Impact of Bilingual CS Education on Student Learning and Engagement in a Data Structures Course

Adalbert Gerald Soosai Raj*
Department of Computer Science and Engineering
University of California, San Diego
gerald@eng.ucsd.edu

Hanqi Zhang†
College of Literature, Science, and the Arts
University of Michigan, Ann Arbor
alisahq@umich.edu

Viren Abhyankar
Department of Computer Science and Engineering
University of California, San Diego
vabhyank@ucsd.edu

Saswati Mukerjee
Department of Information Science and Technology
College of Engineering Guindy, Anna University
msaswati@auist.net

Eda Zhang
Department of Curriculum and Instruction
University of Wisconsin-Madison
rzhang346@wisc.edu

Jim Williams
Department of Computer Sciences
University of Wisconsin-Madison
jimw@cs.wisc.edu

Richard Halverson
Department of Educational Leadership and Policy Analysis
University of Wisconsin-Madison
rich.halverson@wisc.edu

Jignesh M. Patel
Department of Computer Sciences
University of Wisconsin-Madison
jignesh@cs.wisc.edu

ABSTRACT

Learning data structures is hard when taught using a new programming language (e.g., C++), while the students had learned introductory programming in a different language (e.g., C). Learning data structures might even be harder for non-native English speakers, when it is taught in a natural language (e.g., English) that is not the students' native language. We were interested in finding the effect of an instructional design that combines the students' native language (e.g., Tamil) along with English on students' understanding of select topics in a data structures course using C++. We designed an experiment to teach a few data structures (e.g., strings, vectors, maps) in the Standard C++ Library to two groups of undergraduate students in Tamil Nadu, India. We taught the experimental group using English and Tamil (native language of students in Tamil Nadu) and the control group using only English. We conducted a pre-test and a post-test to test students' understanding of programming before and after our intervention. We also conducted an English test to assess their competence in English. We collected data on classroom interaction based on the questions that students asked in lectures during our intervention. We found that teaching data structures using native language and English is no different than teaching data structures using only English. We also found that the native language had an impact on the student engagement and classroom interaction by creating more discussion within the Tamil+English (experimental) classroom when compared to the English-only (control) classroom.

CCS CONCEPTS

• **Social and professional topics → Computer science education**;

KEYWORDS

Bilingual CS Education, Computer Science Education

ACM Reference format:
Adalbert Gerald Soosai Raj, Hanqi Zhang, Viren Abhyankar, Saswati Mukerjee, Eda Zhang, Jim Williams, Richard Halverson, and Jignesh M. Patel. 2019. Impact of Bilingual CS Education on Student Learning and Engagement in a Data Structures Course. In *Proceedings of 19th Koli Calling International Conference on Computing Education Research, Koli, Finland, November 21–24, 2019 (Koli Calling '19),* 10 pages.
https://doi.org/10.1145/3364510.3364518

1 INTRODUCTION

People from all around the world learn computer programming as it is increasingly becoming a much needed and valuable skill in our present technology driven world. Even though people from various native language backgrounds learn programming, the most popular programming languages (e.g., Java, C, Python) are based in English.

*Work done while the first author was a graduate student at UW-Madison
†Work done while the second author was an undergraduate student at UW-Madison

Koli Calling '19, November 21–24, 2019, Koli, Finland

ACM ISBN 978-1-4503-7715-7/19/11…$15.00
https://doi.org/10.1145/3364510.3364518

Also, most of the online resources that are available for learning programming (e.g., Java documentation, Stack overflow, programming tutorials) are primarily in English. Therefore, it becomes necessary for a person to learn English in order to learn programming [14]. The prerequisite of knowing English to learn programming creates a major hurdle for people whose native language is not English and whose proficiency in English is limited [13].

Medium of instruction is the language of communication that is used among teachers and students within a classroom. In many countries where English is not the native language, students have the choice to either study in an English-medium school or a vernacular medium school during their kindergarten through twelfth grade. For instance, in Tamil Nadu, a southern state in India, students may study either in a English-medium school or a Tamil-medium school. Tamil is the native language of people in Tamil Nadu. In an English-medium school, students learn all their subjects in English and learn their native language (Tamil) as a separate subject. On the other hand, in a Tamil-medium school students learn all their subject in Tamil and learn English as a foreign language.

The decision about in which medium of instruction a child studies is made by the child's parents. They primarily make this decision based on their financial income. English-medium education is costly where as Tamil-medium education is free of cost. This cost difference between the two mediums of instruction forces the parents from poor socio-economic status to choose Tamil-medium instruction for their children. Even though learning using the native language has its own advantages, Tami-medium students pursuing STEM courses face huge difficulties as these subjects are primarily taught only in English in college/universities. For instance, most universities in India teach computer science only in English, irrespective of the students' prior medium of instruction.

As Computer Science is taught only in English, this creates a huge barrier for Tamil-medium students and for students who are not competent in English even though they learned in an English medium school, as these students are forced to learn computer science in a language that is not their native language. Also, the subject they are learning (i.e., introductory computer science) is considered to be inherently hard for beginners [3, 28]. Therefore, these students fail the programming/data structures courses and as a result develop inferiority complexes about their programming abilities [17].

Prior work in this area reports mixed results about the effect of the native language in teaching programming. For instance, vernacular medium students learn programming better when taught using their native language (Hindi) when compared to using English [17]. On the other hand, there is no difference between teaching using English versus Tamil and English for non-native English speakers (i.e., both English-medium and vernacular medium students) [25]. These prior works in the area of bilingual CS education have only considered the students' prior programming skills and have ignored the students' English competencies completely. We believe that students' English competencies too might play a role in understanding the effect of dual language instruction for non-native English speakers. In this study, we measure students' English competencies and use it along with the programming pre-test to understand students' prior knowledge before our intervention.

Also, most of the prior work in this area mainly focus on student learning measured using pre and post technical tests [18–20, 24]. By doing so, these studies treat the process of learning as a black box where the input is the language of instruction and the output is the student learning measured using tests. In these prior works, the role of the native language as a facilitator in promoting classroom interaction is not taken into account. In this study, along with measuring student learning using tests, we also conducted classroom observations to note the interactions that happen between teacher and students to get a more holistic view about the role of the native language in teaching and learning computer science.

Our research tries to answer the following two questions:

(1) *What is the impact of bilingual CS education on student learning in an advanced data structures course for non-native English speakers?*
(2) *What is the impact of bilingual CS education on classroom interaction measured using the questions asked by students during the lecture?*

To answer these questions and to better understand the impact of bilingual CS education to teach advanced data structures, we conducted an experiment spanning 4 weeks, in which we taught the C++ Standard Library in Modern C++ [22] using both *Tamil and English* to a group of students (experimental group) whose native language was Tamil. We also taught the same course to another set of students (control group) *only in English*, even though their native language was also Tamil. We used *Tamil along with English* to teach the experimental group since we believe that even though Tamil may help students to better understand programming concepts, they should learn English too since English is the global language for programming and communication [16].

The percentage of native English speakers around the whole world is only 5.3% of the world's population [5]. When including the non-native English speakers who can speak English fluently, this percentage increases to 20% of the world's population [5]. We consider our study to be one of the initial steps taken towards making computing education more accessible to non-native English speakers around the whole world.

2 RELATED WORK

Yogendra Pal [17] conducted a series of controlled experiments in India where he taught programming to two groups of Hindi-medium students (i.e., students who learned through a Hindi-medium of instruction during K-12). He used English to teach the control group and Hindi to teach the experimental group. He found that Hindi-medium students performed better on programming tests when they were taught using Hindi (students' native language) as the medium of instruction when compared to teaching using English as the medium of the instruction. In Pal's studies, the students' prior knowledge in programming was ignored (even though a pre-test was conducted) as student learning was measured using only the post-test scores.

Soosai Raj et. al. studied the effects of bilingual CS education on student learning [24] and sentiments [21] of non-native English speakers. They conducted a pilot study in which they taught a data structures course to two groups of students whose native language was Tamil. They used English to teach the control group and a

mix of Tamil and English to teach the experimental group. They conducted pre- and post-test technical tests and found that bilingual CS education did not have a significant difference with respect to student learning (measured using gain scores). They also found that students expressed positive sentiments about using a bilingual teaching methodology for teaching computer science. Although they measured students' prior knowledge in programming using a pre-test, students' English competencies were ignored. Also, they conducted their pilot study for only a week which is too little time to see any effects in student learning. We consider our study to be an extension of their work, in which we study the effectiveness of bilingual education in teaching and learning Computer Science by conducting a longer study in a similar data structures course and by considering students' English competencies (along with their prior programming knowledge).

Philip Guo conducted a study to understand the barriers that non-native English speakers face while learning to program [13]. He conducted a survey to a programming education website (Python tutor [12]) and analyzed the results. He found that non-native English speakers faced barriers with reading instructional materials, technical communication, reading and writing code, and simultaneously learning English and programming. He also found that they wanted instructional materials that use simplified English and culturally-agnostic code examples. Some non-native English speakers reported that learning programming motivated them to learn English better. Our study aims to reduce one of the difficulties reported by non-native English speakers (i.e., learning English and programming simultaneously) by relaxing the language constraint of using English-only within a computer science classroom.

Sara Vogel et. al. conducted a study to learn about how bilingual learners translanguage when learning computer science [27]. Translanguaging is a process in which bilingual learners use linguistic resources across and beyond multiple languages to learn. They conducted an intervention in which they infused Computational Thinking (CT) [30] in a Spanish-English bilingual language arts class for middle schoolers. Using a qualitative methodology, they claim that students translanguage to engage in specific CT practices. Their results show the importance of allowing students to express themselves using their complete linguistic repertoire while learning computing.

Reestman and Dorn studied the compiler errors of novice programming students from different native language backgrounds [23]. They explored the potential difference in their error distributions relative to those in English speaking backgrounds. They found a statistically significant difference in error distributions between native language groups, but the effect sizes were weak indicating that the differences have little practical significance in terms of guiding either language or instructional design.

The role between native language and learning has also been studied in physics education research (PER). For instance, John Airey [1] conducted a qualitative study in undergraduate physics classes in Sweden. He recorded the lectures including student questions in physics classes that were taught in English and Swedish and used the video recordings to interview students using stimulated recall. He found that students asked fewer questions when taught using English when compared to Swedish. Students also reported that they used their English lectures for mechanical note-taking since they found it difficult to both listen and take notes during the lecture. We consider our study to be an extension of Airey's work to computing education (CER).

3 THEORETICAL BACKGROUND

Translanguaging [29] is a theory from bilingual education and applied linguistics which states that people use the natural languages (e.g., Tamil, English, etc) they know in a fluid manner while they are involved in the process of meaning making [10]. This means that when bilingual learners communicate among each other they tend to use both their primary and secondary languages in such a way that we can't really distinguish one language from the other. Translanguaging argues that the way bilingual or multilingual learners use languages often makes it impossible to categorize their language as either one or the other. In other words, bilingual speakers may use two languages within the same sentence. They may also use one language for explaining the concepts while using technical terms from the other language.

Language plays a primary role in shaping the thought process of students while learning a subject. Translanguaging states that if language barriers are removed and if students and teachers are allowed to communicate in the language of their choice, then it helps to improve the interactions among students and teachers, as they are free to use their complete set of meaning making resources during their classroom interactions. [4].

Translanguaging helps students to use their linguistic resources across and beyond multiple named languages in a computer science classroom [27]. Translanguaging provides us a lens using which we can describe and document the processes of bilingual learners in a bilingual classroom. Translanguaging is relevant to the field of Computer Science since the de facto language of programming is English. Only by allowing students to use their complete set of language resources, we will be able to help learners to better understand code that is primarily in English.

Translanguaging is a relevant theory to apply in the context of Computer Science education at the tertiary levels (i.e., colleges) in India. This is because non-native English speakers learning computer science in India usually communicate with their peers with a mix of both languages (i.e., their primary language – Tamil and their secondary language – English). Most of the interactions that happen outside the classroom often cross these language boundaries in ways students might not even notice. But when these students enter a CS classroom, they are often restricted with respect to the languages they can use within the classroom since most institutions have specific rules about language use within a classroom. Usually, students are expected to communicate using only English while they are in a class. One of the main reasons behind this decision is because colleges and universities want to prepare their students to work in multi-national companies like Google and Microsoft, where the de facto language of official communication is English. So, in order to prepare students for their professional careers, they enforce students and teachers to communicate in English within the classroom. This limits the meaning making capabilities of the students since they are forced to use only their secondary language (i.e., English) within the classroom.

4 METHODOLOGY

In this section, we explain the methodology that we used to conduct our experiment and collect our data.

4.1 Participants

We conducted our study in an Engineering college in Chennai, Tamil Nadu. Our study spanned over 4 weeks during the months of July and August of 2018. The 4 weeks in which our study was conducted correspond to weeks three to six in a 15-week semester. Two groups of second-year students enrolled in two different sections of a data structures course were selected as participants for our study. One group was treated as the control group and the other group was treated as the experimental group. The total number of students in the control and experimental group were 28 and 39 respectively. All students in both the groups participated in our study. Among these students, 2 students in the control group and 5 students in the experimental group studied in a Tamil-medium school during their K-12. All these students have previously taken a programming and basic data structures course in C [15]. The course in which we conducted our intervention was the next course that students take in which they learn advanced data structures using the C++ programming language [26].

4.2 Experimental Design

The experiment was conducted using a *nonrandomized control group pre-test post-test design* [6]. In this design, the participants were not randomly assigned to groups but remained in their pre-assigned groups. This was done as we conducted our experiments as part of regular classes and so were not able to randomize and regroup our participants into two new groups. This increases the external validity of the design by reducing the reactive effects of the experimental procedure [6]. We acknowledge that the problem with this approach is that even if there are any post-test differences between the groups, they may be attributed to characteristic difference between the groups rather than to the intervention. We take this issue into consideration in our result analysis by choosing our statistical models for pre-test post-test comparison very carefully (see Section 5 for more details). The same instructor taught the two groups during our intervention. There was an observer in the classroom who took field notes of the interactions that happened between the students and the teacher during our intervention. The instructor and the observer were part of our group of researchers.

4.3 Experimental Procedure

The following activities were performed with both the English-only (control) group and the Tamil+English (experimental) group as a part of our intervention. There was a pre-test, in-class lectures, and a post-test. The questions in the pre-test and the post-test were in English for both groups.

4.3.1 Pre-test. A pre-test was conducted to determine the students' understanding of the basic concepts in programming. There was a total of 5 questions on the pre-test. The following are the topics for the questions:

(1) C basics
(2) Functions and Recursion
(3) Pointers
(4) Linked Lists
(5) Memory regions (stack vs heap)

The pre-test was conducted for a total of 50 points, 10 points for each question. The pre-test questions were created based on the previous topics that students learnt in the basic programming and data structures courses, in consultation with the instructors who taught those courses.

The above-mentioned topics were tested in the pre-test as they would give us a baseline for understanding the students' prior knowledge about their programming skills before our intervention. The complete pre-test can be found at this link: http://bit.do/cpp_pretest.

4.3.2 English test. A test was conducted to measure the English competencies of the students in the two groups. The students were asked to write an essay in English describing themselves. The students' essays was graded using the following three categories: (1) organization of ideas, (2) grammar, spelling, and punctuation, and (3) style and format. Each of these categories were rated on a four-point scale, namely inadequate (below average) - 0 points, adequate (meets standard) - 1 point, above average (exceeds standard) - 2 points, and exemplary (far exceed standard) - 3 points.

4.3.3 Classroom Lectures. Eight classroom-based lectures, each of 50 minutes duration, were taught for both groups. The C++ Standard Library were taught in those twelve lectures. Topics discussed were: strings, vectors, maps, iterators, algorithms, etc. The same topics were taught to both the groups.

The main differences between the lectures for the two groups were the following: The lectures were taught *only in English* for the students in the control group. Also, the students in the control group were required to communicate with the instructor and their classmates during the lecture only in English.

On the other hand, the lectures were taught using both *English and Tamil* in the experimental group, and the students were free to communicate in any of those two languages, whichever they felt more comfortable with. The instructor used both English and Tamil nearly equally (i.e., 50% time in English - 50% time in Tamil) while teaching the experimental group. The instructor answered the questions during the lecture using the same language (either English or Tamil) in which they were asked.

The instructor used bilingual teaching methods like code-switching [8] and translanguaging [9] for teaching the Tamil+English (experimental) group. The instructor used *code-switching* for switching between Tamil and English in the following way. He used English to introduce a topic, to explain the basic idea behind the topic, and to explain some technical terms (e.g., vectors, maps). He switched to Tamil whenever he felt that a particular topic needed detailed explanation in order to help the students understand the idea in a better way (e.g., How do we insert an element at a particular position in a vector?). The instructor used *translanguaging* as follows. He used Tamil only for oral explanations, discussion, and answering students' questions. He wrote all the content (e.g., topic names, definitions, explanations etc.) on the chalk-board during the lectures only in English.

4.3.4 Post-test. There were three separate post-tests covering the following topics: strings, vectors, and maps. The post-tests were conducted after each topic was covered. There were 6 questions on each topic. All the questions on the post-test were based on the material taught during the classroom-based lectures. The post-test was conducted for a total of 58 points (strings - 18, vectors - 20, maps - 20 points). The complete post-test can be found at this link: http://bit.do/cpp_posttest.

4.3.5 Classroom observations. An observer observed the classes and took field notes of the interactions that happened between the students and the teacher in the classroom. The observer also made notes of the language used for asking the questions. One sample interaction in the experimental group was as follows (Q - question asked by the student; A - answer given by the teacher): *Q: What is the difference between inserting a key value pair using the array syntax versus the insert method in a map? A: The array insert or using the operator [ ] inserts a key value pair only if it not present in the map. If the key already exists, it updates the value for that key with the new value. The insert method only inserts if the key is not present and does nothing if the key is already present in the map.* The full set of questions asked by students in both the groups and the answers given by the instructor can be found here: http://bit.do/cpp_questions

5 RESULTS

The mean of the pre-test scores and the post-test scores for the two groups are shown in Table 1 and Table 2 respectively.

Table 1: Mean of pre-test scores for the two groups

Group	N	Mean	Std. Dev.	Std. Error of Mean
Control	28	46.1	16.9	3.2
Experimental	39	37.1	17.1	2.7

Table 2: Mean of post-test scores for the two groups

Group	N	Mean	Std. Dev.	Std. Error of Mean
Control	28	74.4	16.4	3.1
Experimental	39	62.1	19.9	3.2

The mean of the English writing score (conducted as part of the pre-survey) for the two groups are shown in Table 3.

Table 3: Mean of English test scores for the two groups

Group	N	Mean	Std. Dev.	Std. Error of Mean
Control	28	67.8	17.4	3.3
Experimental	39	60.1	17.4	2.8

5.1 Analysis of Pre-test and English Scores

We compared the pre-test of both groups using an independent samples t-test. The following assumptions for two sample t-test were satisfied:

(1) The Central Limit Theorem (CLT) applies to each sample individually. i.e., pre-test data is normally distributed in both the groups as confirmed by the Quantile-Quantile (Q-Q) plots.
(2) The pre-test scores of both the groups are random samples that are independent of each other.
(3) The variances of the pre-test scores are approximately equal. (i.e., Standard deviation of control group (s1) = 16.9; Standard deviation of experimental group (s2) = 17.1; s2/s1 = 1.01 < 1.5).

We performed an independent samples t-test to compare the pre-test scores between the control group and the experimental group and found *a significant difference* in pre-test scores between the two groups ($t(67) = 2.11$, $df = 65$, $p = 0.03$). This means that the two groups differed significantly with respect to their prior programming knowledge (with an alpha value of 0.05 for statistical significance).

The mean of the English scores for the two groups are shown in Table 3. The mean English score for the control group is higher than that of the experimental group. The assumptions for independent samples t-test (as shown in Section 5.1) were satisfied by the English scores of both the groups. We performed independent samples t-test [11] to compare the English scores between the control group and the experimental group and found *no significant difference* in the English scores between the two groups ($t(67) = 1.79$, $df = 65$, $p = 0.78$).

The post-test scores for the two groups cannot be compared using an independent samples t-test since there is a statistically significant difference between the pre-test scores of the students in the two groups (as shown in Section 5.1). Therefore, we do the following two types of analysis:

(1) Compare the gain scores (i.e., post-test - pre-test) of the two groups using an independent samples t-test (if t-test assumptions are satisfied).
(2) Compare the post-test scores of the two groups using two-way ANalysis of COVAriance (ANCOVA) [11] where the co-variates are the pre-test scores and the English scores.

5.2 Analysis of Gain Scores

The mean of the gain scores for the two groups are shown in Table 4. The mean gain for the control group is higher than that of the experimental group. The assumptions for independent samples t-test (as shown in Section 5.1) were satisfied by the gain scores of both the groups. We performed independent samples t-test [11] to compare the gain scores between the control group and the experimental group and found *no significant difference* in gain scores ($t(67) = 0.72$, $df = 65$, $p = 0.47$).

Table 4: Mean of gain scores for the two groups

Group	N	Mean	Std. Dev.	Std. Error of Mean
Control	28	28.4	18.3	3.5
Experimental	39	25.0	18.9	3.0

5.3 Analysis of Post-test Scores

The pre-test scores of the students in both the groups varied significantly before our intervention as shown in Section 5.1. Therefore, we cannot directly compare the post-test scores between the two groups using independent samples t-test as we did for the gain scores [6]. Instead we use a statistical model to analyze and find if there was a significant effect on the post-test score of a student due to our intervention by controlling for the pre-test score and the English score of the student. We performed ANalysis of COVAriance (ANCOVA) [11] on post-test scores (response or dependent variable) of the two groups with the students' group as the independent (categorial) variable and the pre-test and English scores as the covariates (predictor variable).

The following assumptions of ANCOVA were satisfied for both the pre- and post-test scores:

(1) The pre-test and post-test scores are normally distributed (verified using Q-Q plots).
(2) The homegeneity of variances. The variances of the post-test scores are approximately equal. (i.e., standard deviation of control group (s1) = 16.4; Standard deviation of experimental group (s2) = 19.9; s2/s1 = 1.2 < 1.5). A similar test for pre-test scores is shown in Section 5.1.
(3) The post-test (and pre-test scores) of both the groups are random samples that are independent of each other.

In addition to the above assumptions (similar to independent samples t-test), ANCOVA requires the following assumptions to be satisfied:

(1) The covariate is independent of the treatment effects. This assumption is satisfied as the pre-test scores were collected before our intervention (treatment).
(2) For each independent variable (group), the relationship between the dependent variable (post-test) and the covariates (pre-test and English score) is linear. This assumption is satisfied as shown by the linearity of the regression lines in Figures 1 and 2.
(3) The lines expressing these linear relationships are all parallel (homogeneity of regression slopes). This assumption is satisfied for the covariate English score (see Figure 2) but is violated for the covariate pre-test score (see Figure 1).

We plot a scatter plot in Figure 1 using the pre-test and the post-test scores of the students in both the groups. In this scatter plot, the x-axis represents the pre-test scores of the students and the y-axis represents the post-test scores of the students. The two lines shown in this scatter plot are the regression lines for a particular group that summarizes the relationship between the post-test score and the pre-test score for that group. A similar scatter plot is shown for English score and post-test score in Figure 2.

From Figure 1 we can see that the y-intercept of the control group's regression line (56.81) is higher than the y-intercept of the experimental group's regression line (41.26) which means that for a given pre-test score, a student in the control group had a better post-test score when compared to a student in the experimental group. Similarly, from Figure 2, we can see that the y-intercept of the control group's regression line (49.23) is higher than the y-intercept of the experimental group's regression line (38.55) which means that for a given English score, a student in the control group had a better post-test score when compared to a student in the experimental group.

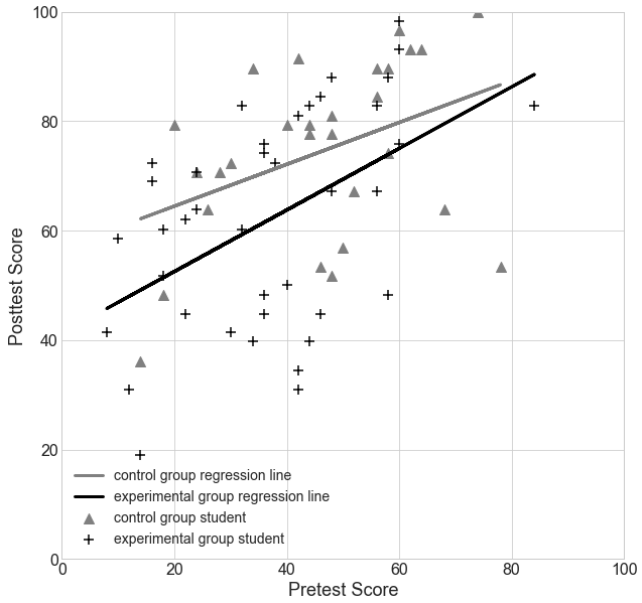

Figure 1: A scatter plot showing each student's pre-test score plotted on the x-axis and the post-test score plotted on the y-axis. The regression lines are also plotted for each group which shows the relationship between the pre-test and post-test scores.

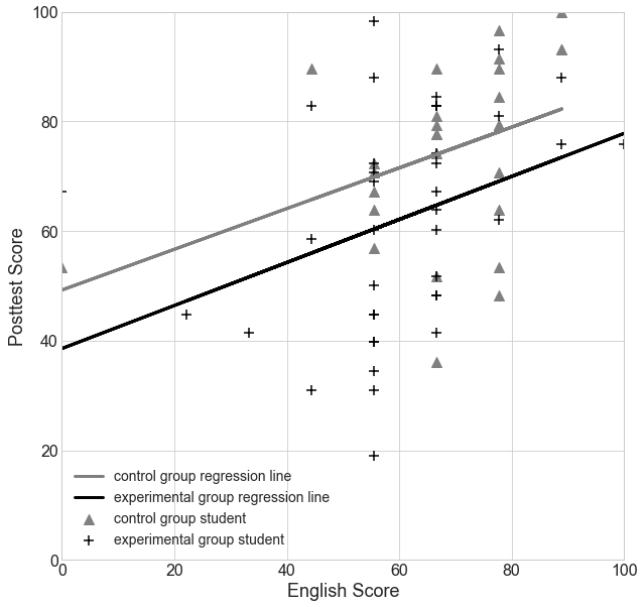

Figure 2: A scatter plot showing each student's English score plotted on the x-axis and the post-test score plotted on the y-axis. The regression lines are also plotted for each group which shows the relationship between the English score and post-test score.

In Figure 2, the regression lines for both the groups have nearly equal slopes (0.37 for the control group and 0.39 for the experimental group). In other words, the two regression lines are almost parallel. On the other hand, the regression lines in Figure 1 are not parallel and have different slopes (0.38 for the control group and 0.56 for the experimental group). This means that we cannot measure the difference between the two group's post-test score by simply factoring out the effect of the pre-test score and the English score (covariates). Instead to find out whether the difference between the two gradients is significant, we should test the interaction of the two covariates (i.e., pre-test score and English score) and the independent variable (i.e., group) to find out if there is any statistically significant difference between the two group with respect to the response variable (i.e., post-test score).

We created a linear model to measure the effect of the interaction between the covariates (pre-test and English score) and the independent variable (group) on the dependent variable (post-test score).

The results from our analysis of covariance are shown in Table 5. The p-values shown in this table are for the interaction of each variable with the response variable (i.e., post-test score).

Table 5: ANCOVA Results

Variable	p-value
Pre-test score and English score	6.488e-06
Group	0.00279
Interaction between the pre-test score + English score and the group	0.48405

The *pre-test score and English score of a student* had a *significant effect on the post-test score* of the student after controlling for the effect of the student's group (p-value = 6.488e-06). This means that irrespective of the group that a student belongs to, generally, if a student did well on the pre-test and the English test then that student also did well on the post-test.

The *group of a student* had a *significant effect on the post-test score* of the student after controlling for the effect of the student's pre-test score and English score (p-value = 0.00279). This is due to the fact that even before our intervention, the control group was better than the experimental group with respect to the knowledge of programming that was tested based on the pre-test score.

The interaction between the *pre-test score + English score and the group of a student* has *no significant effect on the post-test score* of that student. In other words, the post-test scores of two students with the same pre-test score and English score were not statistically significant based on the student's group. This means that the post-test scores of the two groups are not statistically significant due to our intervention even after considering the effects of the pre-test and English score between the two groups.

5.4 Analysis of Student Questions

The number of in-class questions asked by students in both groups is shown in Table 6. The total number of in-class questions asked by students in the Tamil+English (experimental) group (# questions = 17) is slightly more than twice the number of in-class questions asked by students in the English-only (control) group (# questions = 8). All these questions were asked by different students during the lecture. In other words, no student asked more than one question.

Table 6: Number of in-class questions asked by the students in the two groups

Group	Total questions	# English	# Tamil
Control	8	8	0
Experimental	17	4	13

In the experimental group, the number of questions asked in Tamil (# questions = 13) were more than the number of questions asked in English (# questions = 4) by a factor of 3 (i.e., a ratio of 3:1 in favor of Tamil). In the control group, there weren't any question that were asked in Tamil since the students were required to communicate only in English.

The split-up of the number of questions that were asked during each of the twelve lectures in both the groups is shown in Figure 3.

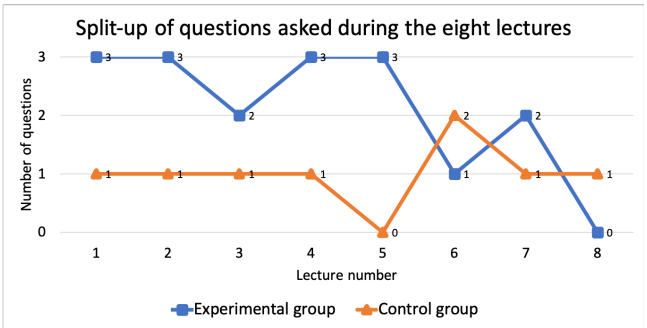

Figure 3: Split-up of student questions by lecture number in both groups.

We classify the questions asked by the students in both the groups into the following four types of knowledge dimensions based on revised Bloom's taxonomy [2]:

(1) Factual knowledge: The basic elements a student must know to be acquainted with a discipline or solve problems in it. (e.g., What is the difference between size() and length() methods in a string?)
(2) Conceptual knowledge: The interrelationships among the basic elements within a larger structure that enable them to function together. (e.g., Why does the method at() does bounds checking but the array type of access doesn't?)
(3) Procedural knowledge: How to do something, methods of inquiry, and criteria for using skills, algorithms, techniques, and methods. (e.g., Why do we need an iterator to insert an element in a vector using the insert method?)
(4) Metacognitive knowledge: Knowledge of cognition in general as well as awareness and knowledge of one's own cognition (e.g., Why do we need to pass v.begin() and v.end() to functions like reverse() and sort() instead of passing in just the vector?)

We used revised Bloom's taxonomy [2] as it provided a way for classifying our questions into one of the different knowledge dimension namely factual, conceptual, procedural, and metacognitive.

Two researchers from our team classified the questions into one of these four knowledge dimensions using a deductive content

analysis [7]. The two researchers agreed upon the classification for 92% of questions (i.e., 23 out of 25 questions). For the two questions they did not agree, our researchers used discussion as a way to converge to an agreement.

The number of questions that are classified across the four different knowledge dimensions is shown in Table 7.

Table 7: Classification of questions across 4 knowledge domains

Group	Factual	Conceptual	Procedural	Metacog.
Ctrl	4	2	1	1
Exp	7	6	1	3

There were four factual questions in the English-only (control) group and seven factual questions in the Tamil+English (experimental) group. There were two conceptual questions in the English-only group while there were six conceptual questions in the Tamil+English group. There was only one procedural question that was asked in both the groups. There was one metacognitive question in the English-only group while there were three such questions in the Tamil+English group.

The questions across the four knowledge domains asked by the students in the experimental group are classified based on the language (Tamil or English) that was used to ask the question and the results are summarized in Table 8.

Table 8: Classification of questions across the four knowledge domains from the Tamil+English (experimental) group based on the language

Domain	English	Tamil
Factual	2	5
Conceptual	2	4
Procedural	0	1
Metacognitive	0	3

There were two factual questions in English and five in Tamil. Four conceptual questions were asked in Tamil while two were asked in English. All procedural and metacognitive questions were asked in Tamil.

Even though there were not as many questions in the control group as there were in the experimental group during the lectures, the number of students who asked questions on a one-on-one basis after class were more in the English-only (control) group than in the Tamil and English (experimental) group. The number of students who asked questions after class on a one-on-one basis in the English-only group and Tamil+English group are 12 and 4 respectively. The comparison between the number of in-class questions and the out-of-class questions between the two groups is shown in Figure 4. The out-of-class questions were not included in our classification of questions shown in Tables 6, 7, and 8. The out-of-class questions were asked by the students when the instructor was packing up and walking out of the class. Therefore, our observer was not able to record these questions for analysis.

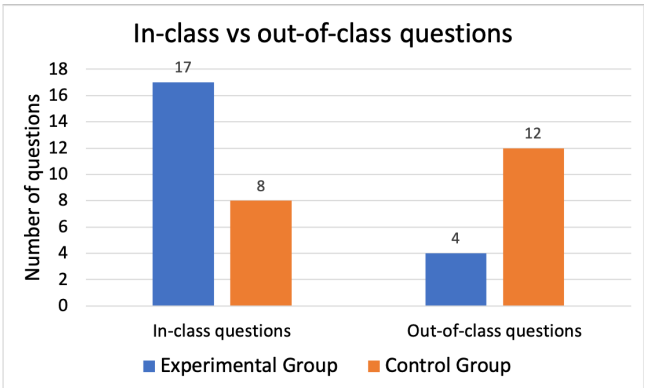

Figure 4: Comparison of the number of in-class and out-of-class student questions in the two groups

6 DISCUSSION

6.1 Interpretation of Results

Our study tried to find if using the native language (Tamil) along with English for teaching an advanced data structures course using C++ had any effect on students' learning of programming and classroom interaction when compared to using only English. We measured the student learning in terms of gain scores and post-test scores. We measured the classroom interaction based on the questions asked by the students during the lectures.

The difference between the two groups with respect to the gain score is not statistically significant (see Section 5.2). Also, there was no significant difference between the post-test scores between the two groups when controlling for the pre-test and English scores as the covariates. This shows that teaching programming using Tamil and English is no different than teaching programming using only English with respect to student learning measured by means of test scores.

Our finding on the effect of native language on student learning in an advanced data structures course matches the findings of Soosai Raj et. al. [24] which found that the use of the native language (Tamil) did not have a significant positive impact on student learning in a data structures course where linked lists was taught using the C programming language. We consider our findings to be a validation of the previous results even after considering students' English proficiency.

Our findings add more value to the findings of Pal and Iyer [17] which suggests that the native language (Hindi) had a significant positive effect on students who did their schooling in a Hindi-medium school. An important difference between our study and Pal's study is that Pal et. al. conducted their study in a first programming course while we conducted our study in data structures course. Therefore, even though the effect of the native language was different in both these studies, the target audience were different with respect to their prior programming experience and native language (Tamil vs Hindi).

Based on the classroom observations we conducted as part of our study, we collected data on the questions that were asked in the classroom, during our intervention, in both the groups. We found that giving students the freedom to speak using either their native language (Tamil) or English increased the classroom interaction that happened during the lectures (measured using the questions that were asked by the students in-class) in the Tamil+English

(experimental) group when compared to the English-only (control) group (see Table 6).

We classified the questions along four knowledge dimensions based on the revised Bloom's taxonomy [2] and found that the students in the Tamil+English (experimental) group asked more questions about conceptual, procedural, and meta-cognitive knowledge when compared to number of factual questions. The students in the English-only (control) group asked equal number of questions in the factual when compared to conceptual, procedural, and factual combined. We also found that nearly 76% of the questions that students asked in the Tamil+English group were in Tamil. Our results on analyzing the student questions shows that the use of the native language had an impact on the number and the quality of questions that non-native English speakers ask in a CS classroom.

An alternative interpretation of our results is that the students in the Tamil+English (experimental) group are better students when compared to the students in the English-only (control) group. Therefore, the questions that these students asked might not be due to the native language but instead due to their inherent curiosity. We argue that this might not be the case since the students in the English-only group performed better on the pre-test and the English test when compared to the students in the English+Tamil group.

Our results on the classroom interaction matches those observed by Airey [1] while teaching physics using English and Swedish. For example, Airey found that English is a barrier for students to ask questions during a lecture because students do not want to be embarrassed before their peers by asking a question in English. We found that after the lecture, the number of students who asked questions on a one-on-one basis was higher in the English-only group when compared to the English+Tamil group. We believe that this may be due to the fact that English is a barrier for non-native English speakers for asking questions during the class.

6.2 Limitations and Future Work

One of the major limitations of our study is that the course we performed our intervention was not the *first* course that students take when they learn programming. Therefore, we believe that there is a good chance that many students who might have needed our intervention the most (e.g., Tamil-medium students), may not even be present in our study since they could have dropped out immediately after their first CS course.

We used a *nonrandomized control group pre-test post-test design* because we conducted our intervention as part of regular classes and so we were not able to randomize students into two groups. So, we used the two pre-formed groups as our experimental and control groups. As a consequence, the two groups varied significantly with respect to their prior knowledge in programming and data structures before the start of our intervention. Although we have taken this pre-test difference into consideration in our analysis of results, we acknowledge that the results would have been more reliable if these initial differences didn't exist among these groups. This is a major limitation of our study. To minimize the effects due to the initial differences among the two groups, as a part of our future work, we plan to conduct more controlled experiments with a *randomized control group pre-test post-test design* [6] to better understand the effects of the native language for learning data structures.

Another limitation with our study is the way we measured students' English competencies. We used a writing test where the students wrote an essay about themselves. An essay type of question only measured the writing competency of the student which might not be a holistic assessment of their English competencies. Therefore, we plan to conduct more comprehensive English assessments in the future, in which we measure multiple skills like reading, listening, and speaking along with the students' writing skills.

7 CONCLUSION

We found that teaching data structures in C++ using a bilingual teaching methodology (in Tamil and English) is no different than teaching using only English with respect to student learning. We also found that the native language had a positive impact on the classroom interaction, measured using the quantity and the quality of student questions that were asked during the class. We conclude that more studies should be conducted in bilingual CS education, across different CS courses and using different natural languages (e.g., Malayalam) to truly understand and benefit from the role that a natural language plays in learning computer science education.

REFERENCES

[1] John Airey. 2009. *Science, language, and literacy: Case studies of learning in Swedish university physics.* Ph.D. Dissertation. Acta Universitatis Upsaliensis.
[2] Lorin W Anderson, David R Krathwohl, Peter W Airasian, Kathleen A Cruikshank, Richard E Mayer, Paul R Pintrich, James Raths, and Merlin C Wittrock. 2001. A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, abridged edition. *White Plains, NY: Longman* (2001).
[3] Jens Bennedsen and Michael E Caspersen. 2007. Failure rates in introductory programming. *AcM SIGcSE Bulletin* 39, 2 (2007), 32–36.
[4] Angela Creese and Adrian Blackledge. 2010. Translanguaging in the bilingual classroom: A pedagogy for learning and teaching? *The modern language journal* 94, 1 (2010), 103–115.
[5] David Crystal. 2006. *English worldwide.* Cambridge University Press, 420âĂŞ439. https://doi.org/10.1017/CBO9780511791154.010
[6] Dimiter M Dimitrov and Phillip D Rumrill Jr. 2003. Pretest-posttest designs and measurement of change. *Work* (2003).
[7] Satu Elo and Helvi Kyngäs. 2008. The qualitative content analysis process. *Journal of advanced nursing* 62, 1 (2008), 107–115.
[8] Jennifer R Fennema-Bloom. 2009. Code-scaffolding: A pedagogic code-switching technique for bilingual content instruction. *Journal of Education* (2009).
[9] Ofelia García and Claire E Sylvan. 2011. Pedagogies and practices in multilingual classrooms: Singularities in pluralities. *The Modern Language Journal* (2011).
[10] Ofelia García and Li Wei. 2014. Translanguaging. *The Encyclopedia of Applied Linguistics* (2014), 1–7.
[11] Gene V Glass and Kenneth D Hopkins. 1970. *Statistical methods in education and psychology.* Prentice-Hall Englewood Cliffs, NJ.
[12] Philip J Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education.* ACM, 579–584.
[13] Philip J Guo. 2018. Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* ACM, 396.
[14] Scott Hanselman. 2008. Do you have to know English to be a Programmer? https://www.hanselman.com/blog/DoYouHaveToKnowEnglishToBeAProgrammer.aspx. (2008). [Online; accessed 12-August-2019].
[15] Brian W Kernighan and Dennis M Ritchie. 2006. The C Programming Language. (2006).
[16] Thomas Andrew Kirkpatrick. 2011. Internationalization or Englishization: Medium of instruction in today's universities. (2011).
[17] Yogendra Pal. 2016. *A Framework for Scaffolding to Teach Programming to Vernacular Medium Learners.* Ph.D. Dissertation. IIT, Bombay.
[18] Yogendra Pal and Sridhar Iyer. 2012. Comparison of English versus Hindi Medium Students for Programming Abilities Acquired through Video-Based Instruction. In *T4E.* IEEE.
[19] Yogendra Pal and Sridhar Iyer. 2015. Classroom Versus Screencast for Native Language Learners: Effect of Medium of Instruction on Knowledge of Programming. In *ITiCSE.* ACM.

[20] Yogendra Pal and Sridhar Iyer. 2015. Effect of medium of instruction on programming ability acquired through screencast. In *LaTiCE*. IEEE.
[21] Adalbert Gerald Soosai Raj, Kasama Ketsuriyonk, Jignesh M Patel, and Richard Halverson. 2017. What Do Students Feel about Learning Programming Using Both English and Their Native Language?. In *LaTICE 2017*. IEEE.
[22] Adalbert Gerald Soosai Raj, Varun Naik, Jignesh M Patel, and Richard Halverson. 2018. How to teach modern C++ to someone who already knows programming?. In *Proceedings of the 20th Australasian Computing Education Conference*. ACM, 97–104.
[23] Kyle Reestman and Brian Dorn. 2019. Native Language's Effect on Java Compiler Errors. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ACM, 249–257.
[24] Adalbert Gerald Soosai Raj, Kasama Ketsuriyonk, Jignesh M Patel, and Richard Halverson. 2018. Does Native Language Play a Role in Learning a Programming Language?. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 417–422.
[25] Adalbert Gerald Soosai Raj, Eda Zhang, Saswati Mukherjee, Jim Williams, Richard Halverson, and Jignesh M Patel. 2019. Effect of Native Language on Student Learning and Classroom Interaction in an Operating Systems Course. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 499–505.
[26] Bjarne Stroustrup. 2000. *The C++ programming language*. Pearson Education India.
[27] Sara Vogel, Christopher Hoadley, Laura Ascenzi-Moreno, and Kate Menken. 2019. The Role of Translanguaging in Computational Literacies: Documenting Middle School Bilinguals' Practices in Computer Science Integrated Units. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 1164–1170.
[28] Christopher Watson and Frederick WB Li. 2014. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 39–44.
[29] Li Wei. 2017. Translanguaging as a practical theory of language. *Applied Linguistics* 39, 1 (2017), 9–30.
[30] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.