Understanding the Transfer of Knowledge in Introductory Programming Classes

Michael Knox, Charlotte Tang, Halil Bisgin, Murali Mani, Suleyman Uludag
Department of Computer Science, University of Michigan-Flint
Email: {knoxm, tcharlot, bisgin, mmani, uludag}@umich.edu

***Abstract*—Computing has become an essential field of study, with programming as its core component. However, pedagogical aspects of teaching programming remain under-explored and poorly understood. The selection of a student's first programming language has sparked considerable debate within the community, often referred to as a "culture war", and presents curricular challenges that extend to subsequent language choices.**

This research full paper investigates the transitions between different programming languages and measures students' learning experiences as they progress through a single language. We utilize Mindshift Learning Theory (MLT) to examine the transfer of knowledge across programming languages as students go through courses in a pre-requisite chain in our curriculum. Our study focuses on three aspects: novelty (encountering new, unique concepts), change (differently applying previously exposed topics), and carryover (directly transferring concepts to new contexts).

Our methodology includes an online survey of current and past students transitioning through these courses. We aim to understand how perceived transitions impact mastery and confidence in programming languages. We designed the survey questions, identified target classes, and implemented incentive mechanisms to ensure a statistically significant response rate.

Our results are partially aligned with earlier findings in the literature, which correlated carryover, changed, and novel concepts with knowledge scores of object oriented (OO) developers, despite a totally different demographic. Namely, our student participants transitioning to C/C++ (procedural programming) and OO C++ mostly have high, moderate, and low knowledge scores on the programming concepts they perceive as carryover, novel, and changed respectively. On the other hand, those transitioning to OO Java have higher scores for the changed concepts.

***Index Terms*—Knowledge Transfer, Transitioning, Programming Languages, Mindshift Learning Theory, College Students**

I. Introduction

Computing and computational thinking are becoming an indispensable facilitator of opening up new possibilities and horizons for all disciplines of study with profound potential benefits to our societies. Computing, where p**R**ogramming is a fundamental component, is becoming—or has already become—the fourth **R** of the basic literacy and education, alongside **R**eading, w**R**iting, a**R**ithmetic[1] [2] and even earlier by Wing in [3]. Alan Perlis, the 1962 ACM Turing Award winner, predicted that programming should be part of a well-rounded education [4], and this idea is gaining increased support within different academic circles.

The swift integration of generative artificial intelligence (AI) within technological frameworks accentuates the imperative for comprehensive computing education with programming at the core. As articulated in [3], computational thinking and foundational knowledge in computer science are crucial for all students in today's technologically ubiquitous society. This urgency is underscored by the pervasive influence of AI across various sectors, highlighting a significant shift in the required skill sets for the 21st-century workforce. Educational systems must evolve to prepare students for these emerging challenges by embedding robust computer science curricula that not only keep pace with technological advancements but also equip students with critical problem-solving skills. Such an alignment between educational frameworks and modern technological demands is essential for developing future-ready students who are proficient in navigating and contributing to increasingly digital landscapes.

According to the US Department of Labor, Occupational Outlook Handbook [5], employment in the field of computer and information technology is projected to grow *much faster than the average* rate of all other occupations from 2022 to 2032 with about 377,500 expected job openings.

It is then not surprising to see an all-time high of enrollment in computing [6], with the general expectation that it will remain high [7]. Computer and Information Science degree completers, as depicted from [8] Figure 1, shows a significant ever-upward trajectory, where a term coined in [7] refers to it as *Generation CS*. In spring 2023, computer science undergraduate programs at four-year institutions experienced their highest growth rate in three years, increased by 11.6%, which corresponds to an additional 62,000 students [10].

It is one thing to get students flocking into computing programs and classes, but it is quite a different challenge to keep them there. With no consensus on the definition of *Retention* in computing, its inconsistent use and potential remedies across institutions are inevitable [11]. Figure 2 shows the Bachelor's student retention percentages by different computing degrees from 2017-18 through 2021-22 academic years [8], with an obvious room for improvement. ACM's *Retention in Computer Science Undergraduate Programs in the U.S.* [11] recommends further research to gain a more detailed and deeper understanding of the dynamics of attrition

[1]The fourth **R** is noted as algo**R**ithmic or computational thinking by Cathy Davidson in [1].

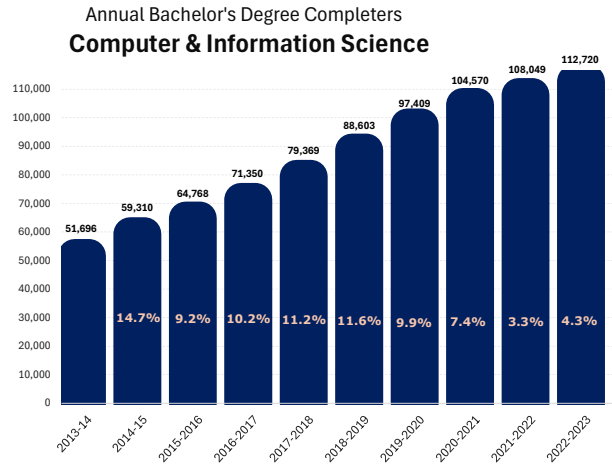

Fig. 1: Annual Bachelor's Degree Completers in Computer and Information Sciences. The number at the top of the bars show the total number of degree completions while the percentages in the middle show the increase from the previous year [9].

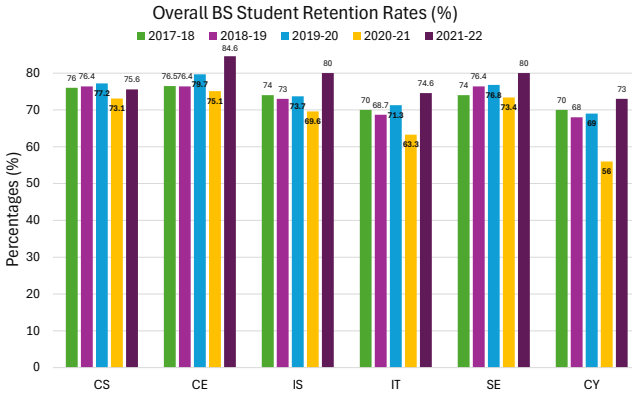

Fig. 2: Overall Bachelor's student retention percentages by computing discipline from 2017-18 through 2021-22 [8]. CS: Computer Science, CE: Computer Engineering, IS Information Systems, IT: Information Technology, SE: Software Engineering, CY: Cybersecurity.

and retention, to identify the factors contributing to decreased retention, and to develop strategies to address these factors.

It is at this juncture that we are looking into the topic of a student's introductory programming language [12], [13] to get some understanding of the transfer of knowledge across programming languages and classes. Yet, many pedagogical and educational dimensions of how to teach coding have been understudied, and hence not well understood [14]. While the choice of the first programming language has been a subject of controversy within the community [15], [16] with different views [17]–[19], often referred to as a "culture war" [20], it gives rise to curricular challenges that extend to the choice of subsequent languages.

We tap into the vast and rich literature of educational psychology, provide a review and taxonomy of the main theoretical frameworks, and then use one of them, Mindshift Learning Theory (MLT) [21], to seek insight about transfer of knowledge from one programming language/level/class to another in a typical computing curriculum. Our methodology includes an online survey of students who are currently transitioning across our programming, C/C++, OO C++, and OO Java, courses, as well as students who made these transitions in the past. We study how the student perceived transitions impact student mastery and confidence in programming languages. The data were analyzed to build models for examining the relationship between novelty and knowledge score.

The rest of the paper is organized as follows: Section II provides an overview of the related work on transfer of knowledge. The details of the survey study is explained in Section III. The results of our study together with its statistical summary and discussions are described in Sections IV and V. Concluding remarks close our paper in Section VI.

II. Related Work

Transfer of knowledge, learning, skills, and experience (ToK)[2] is crucial in today's globalized, highly competitive, and complex world. In addition to its critical role in classroom and academia, ToK is a key factor in training, reskilling, upskilling, and retooling in the corporate world. Yet, it has been very elusive and challenging to study in part due to the following reasons: (1) intricate and complex nature of ToK, (2) ToK's multidimensionality of factors contributing to a successful transfer, and (3) limited causal understanding of the instructional methodologies to facilitate, initiate, or effect a successful transfer.

Figure 3 shows various different conceptualizations of ToK in traditional and contemporary views, taxonomies, and models together with a new integrated model [22]. The first official study of knowledge transfer was conducted by Thorndike and Woodworth [23], [24] which led to a wealth of follow-up studies. These studies were in the traditional *behaviorist* view with a focus on common elements shared between two contexts. The *cognitivist* view [25]–[27] considers the concepts, theories, and general cognitive abilities surrounding knowledge transfer. The *situated* view [28], [29] emphasizes the social and interactive nature of learning characterized by sociocultural, contextual, and situational features. Traditional and contemporary models and taxonomies of ToK are also listed in Figure 3 to explain different aspects of transfer in a variety of different contexts.

A more detailed illustration of the Integrated ToK Model is also given in Figure 4 where the dynamic and interactive nature of transfer is conceptualized along four distinct dimensions: (a) pedagogical, (b) task, (c) context, and (d) personal [22].

One cognitivist ToK approach is the *Mindshift Learning Theory* (MLT). Mindset [21] refers to distinct viewpoints that determines how individuals interact with events or view reality. Mindset change is defined as *mindshift*, which also

[2]We will use the concise acronym *ToK* to refer to the transfer of knowledge, skills, experience, and learning in the broadest sense throughout this study.

Transfer of Knowledge/Learning (ToK) Taxonomies

- **(1) Historical View of ToK**
  - (a) Behaviorist View [23], [24]
  - (b) Cognitivist View [25]-[27]
  - (c) Situated View [28], [29]
- **(2) Traditional models & taxonomies of ToK**
  - (a) Near vs Far [30]
  - (b) High- vs Low-Road Transfer [31], [32]
  - (c) Positive vs negative Transfer [33]
  - (d) Literal vs Figurative [34]
  - (e) Vertical vs Lateral Transfer [35]
  - (f) Specific vs Nonspecific Transfer [36]
  - (g) Common Elements Model [24]
  - (h) General Principle Model
  - (i) Information Processing Model [25]
  - (j) Haskel's taxonomies of ToK [37]
- **(3) Contemporary models & taxonomies of ToK**
  - (a) Taxonomy for Far Transfer [38]
  - (b) Model of the Transfer Process [39]
  - (c) Model of Dynamic Transfer [40]
  - (d) Vertical vs Horizontal Transfer [40]
  - (e) Preparation for future learning [41]
  - (f) Actor-oriented transfer [42]
  - (g) Successful transfer of learning model [43]
- **(4) Integrated ToK model [22].**

Fig. 3: Historical view, traditional and contemporary models, and taxonomies of ToK.

corresponds to punctuated change [44] and second degree order change [45], [46]. Originally developed in [47], MLT attempts to break down a human's approach to learning concepts into three distinct categories, *novel*, *discrepant*, and *deliberative* [47]. However, the model has been reworked by Armstrong and Hardgrave to better fit the programming paradigm [21]. In their work, they examined professional programmers' transition from traditional to object-oriented programming (OOP). Instead of the three learning principles defined by the original MLT creators, they defined their own version specific for software development: novel, changed, and carryover. Their study concluded the professionals had an easier time learning concepts they perceived as novel or carryover versus concepts they perceived as not novel.

Our focus in this study is on MLT in an attempt to gain more insight into its relevance and usefulness in today's highly

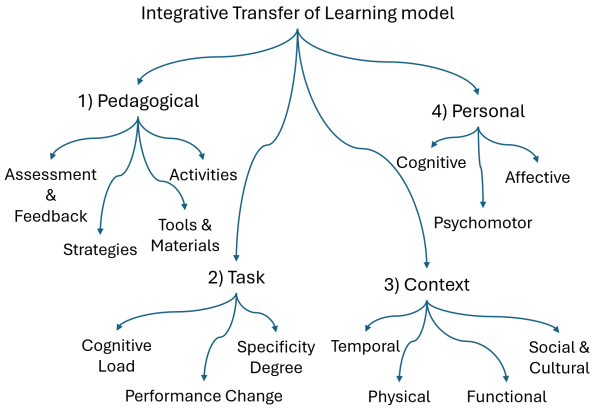

Fig. 4: Integrative Transfer of Learning model [22].

dynamic, complex, and different operating conditions than the original work in [21] from 17 years ago.

III. Methodology

The primary objective of our survey study was to better understand how college students transition from one programming language to another, in order to identify ways to improve computing curriculum and student retention. We begin by describing our participants, then briefly present the questionnaire used in this research, followed by the procedure and data analysis.

A. Participants

We recruited students of age 18+ at a university in mid-western U.S., who were taking or have taken at least two programming courses in Python, C/C++, OO C++, and OO Java. At our study site, Python is a prerequisite for C/C++, which in turn is a prerequisite for OO C++ and OO Java. Students may select either OO programming course. In this paper, we use C/C++ to refer to our procedural C++ programming course, which does not cover OO aspects. We solicited help from computer science faculty who were teaching C/C++, OO C++, OO Java, data structures, data mining, algorithms as well as computer ethics courses. Note that data structures, data mining and algorithms courses have a pre-requisite of either OO C++ or OO Java. The computer ethics course requires students to be of junior standing. The survey link was then made available to the students in these classes near the end of a semester. Participants were offered an opportunity to enroll in a draw with a one-in-four chance for a gift card. Participants were allowed to complete the survey at a location of their choice, and at their own pace. The online questionnaire was delivered through Qualtrics, the authors' affiliated university's officially supported survey system. Fifty-four surveys were received, but 26 had to be discarded because the respondents completed only the demographic questions. The respondents' demographic information is shown in Table I.

B. Method

An in-depth review of literature on knowledge transfer in the cognitive psychology domain was first conducted prior to

TABLE I: Respondents' Demographics.

Age	Mean	23.25
	SD	5.4
Gender	Male	14
	Female	10
	Non-binary	2
	Prefer not to say	2
Race	White/Caucasian	20
	Black/African American	1
	Asian	3
	Mixed	3
	Prefer not to say	1
Native language	English	27
	Vietnamese	1
Major	Computer Science	14
	Computer Information System	2
	Software Engineering	1
	Cybersecurity	2
	Data Analytics	1
	Information Tech & Informatics	2
	Mathematics	3
	Actuarial Mathematics	2
	Environmental Engineering	1
Year of College	Freshman	4
	Sophomore	9
	Junior	9
	Senior	6
First Learn to Program	High School	16
	Community College	2
	4-year College	8
	Don't remember	2
Programming Proficiency	Fundamental Awareness	5
	Novice	9
	Intermediate	13
	Advanced	1
	Expert	0

developing the survey instrument, as detailed in Section II. Among them, we found Mindshift Learning Theory, which proposes that "the degree of perceived novelty of the fundamental concepts that characterize the new mindset will impact learning" [21], most applicable to our study in which we investigated knowledge transfer across multiple programming languages. This theory was thus used to guide the development of our survey. In addition, our survey questions were inspired by the questionnaire used in [21], also guided by the same theory, which investigated the transitioning from traditional to OO programming language while our research explored three different transitions: Python to C/C++, C/C++ to OO C++, and C/C++ to OO Java.

The survey asked primarily closed-ended questions to gather quantitative data, focusing on respondents' experience in transitioning to a new programming language. It consisted of one block of demographic questions and three blocks of focused programming knowledge questions on C/C++, OO C++, and OO Java respectively. All respondents were required to complete the demographic questions (see Table I). The first demographic question asked for respondents' informed consent; the last asked which programming languages they were taking or have taken, the responses to which would determine the programming block(s) the respondents would be directed to.

All three programming language blocks consisted of three sections. The first section in each block asked about their perceived novelty (0-100%) regarding each of nine key concepts relevant to specific programming language. Specifically, the nine concepts for C/C++ were variable and data types, assignment, functions, conditionals, loops/iterations, passing parameters to functions, pointers; the nine concepts for OO C++ and OO Java were the same: polymorphism, object, inheritance, class, encapsulation, attribute, method/function, exception, and abstraction. The second section consists of 27 (3 per concept) knowledge questions (either multiple choice or true/false) on the programming language. The last section is a 7-point Likert scale question asking about their perceived difficulty, from extremely difficult to extremely easy, of each of the nine concepts.

C. Data Analysis

Our analysis mainly relies on the relationship between novelty and knowledge score for a given programming concept. More specifically, we aim to examine how the degree of perceived novelty, which can be broken down to carryover, changed, and novel concepts [21], could impact learning. To this end, we follow a similar approach as in [21] which hypothesized that carryover and novel concepts would have high knowledge score whereas changed concepts tended to have lower knowledge score. The authors formalized and tested this hypothesis by fitting a U-shaped (curvilinear) model to developers' degree of perceived novelty for OO concepts to predict their concept knowledge score. They reported the quadratic model shown in Figure 5 in which vertical lines at 25% and 75% marks the borders between three categories of novelty, i.e., carryover, changed, and novel.

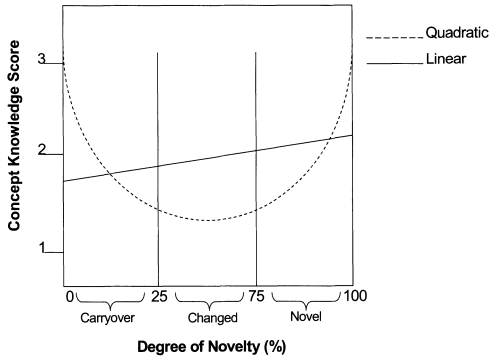

Fig. 5: Regression plots from [21].

Although the authors claimed that a quadratic model was more capable of capturing the non-linear relationship between novelty and score for a given concept, they still kept the linear regression model in Figure 5. Following the same approach, for every transition we have in our survey, we fitted both linear and non-linear models to reveal any potential relationships in our student cohort. Specifically, we built the following models expressed in Equations 1 and 2, where α, β_1, β_2 represent the constant term and the coefficients we want to fit to our data points. While our goal was to map novelty levels to different

regions of a second degree polynomial and associate them with knowledge scores, we also aimed to utilize linear models to discern the overall trend.

$$Score = \alpha + \beta_1 \times Novelty + \beta_2 \times Novelty^2 \quad (1)$$

$$Score = \alpha + \beta_1 \times Novelty \quad (2)$$

Before proceeding to the data modeling stage, we calculated concept knowledge scores by counting the number of correct answers for each concept, since every concept had three questions. This led to concept knowledge scores ranging from 0 to 3. Our degree of perceived novelty, however, ranged from 0 to 100 on a continuous scale. It is worth noting that while these concepts were the same for OO languages, they differed for the C/C++ questionnaire which let us perform a multi-faceted analysis in contrast to [21] where only the transition to OO development was studied among professionals as opposed to students in our study.

IV. Results

As knowledge transfer can manifest itself in different forms and be impacted by the degree of perceived novelty among many other factors suggested by the Mindshift Learning Theory, we laid out different transition scenarios from our home institution. While Armstrong et al. [21] investigated a similar phenomenon from conventional programming to OO development, we investigated transitions between courses that are part of a prerequisite chain in our programming curriculum. We analyzed responses from survey participants transitioning to C/C++ (procedural programming), OO C++, and OO Java.

A. Statistical summary

The survey allowed participants to take multiple knowledge score tests in the case where they had taken both C/C++ courses or a C/C++ course and an OO course. We also had students who have taken both OO courses due to their standing or transfer status from another institution. Hence, we had 28, 22, and 20 students who completed the knowledge block for C/C++, OO C++, and OO Java, respectively.

Having three questions for each concept, we recorded the number of correct answers for each concept as mentioned earlier. By doing so, we quantified the concept knowledge score with a single value, which was paired with the degree of perceived novelty for every concept. Since every participant responded for nine different concepts, we obtained nine pairs of observations, (Novelty, Score), for each record in our survey. In total, we had 630 observations (m) out of which C/C++, OO C++, and OO Java had their shares 252, 198, and 180, respectively.

In Table II we present a statistical summary of each group for both their concept knowledge scores and novelty perceptions. As it can be seen from the table, C/C++ takers have lower median and maximum values for their degree of perceived novelty when compared to other groups. Their average concept knowledge is also slightly greater than OO C++ takers and much higher than OO Java participants. Meanwhile, OO Java takers expressed a higher degree of perceived novelty towards the concepts given.

TABLE II: Statistical summary of participant groups.

	C/C++ (m=252)		**OO C++** (m=198)		**OO Java** (m=180)	
	Novelty	**Score**	**Novelty**	**Score**	**Novelty**	**Score**
min	0	0	0	0	0	0
1^{st} Qu.	10	1	35.25	1	31.75	1
median	36	2	62.5	2	60.5	1
mean	40.32	2.1	59.84	1.9	60.44	1.53
3^{st} Qu.	70	3	81.75	3	95.25	3
max	100	3	100	3	100	3

B. Curvilinear relationship

We first visited the hypothesis as to whether concept score would follow a curvilinear relationship with the degree of perceived novelty associated with the same concept. To this end, we fit a quadratic model, i.e., second degree polynomial to our observations. We fit a different model for each group and their aggregated data, which resulted in two additional models for OO courses combined and all 630 observations we got. While our goal was to see how carryover, changed, and novel concepts were aligned with an earlier assumption [21] by manifesting themselves on a curvilinear relationship, we also presented the linear relationship to capture an overall trend. Accordingly, we built models in the Equation 1 and 2 for which we present their components in Table III where "Poly" refers to polynomial or the quadratic model. Although these models had small R^2 values, they all were found to have their predictors statistically significant.

TABLE III: Fitted model features for each cohort.

	C/C++		**OO C++**		**OO Java**	
	Linear	**Poly**	**Linear**	**Poly**	**Linear**	**Poly**
α	2.251	2.091	2.341	1.919	1.147	1.533
β_1	-0.004	-2.147	-0.007	-2.798	0.006	2.875
β_2	-	2.733	-	1.667	-	-2.463
R^2	0.020	0.052	0.038	0.051	0.039	0.068
$p-value$	0.025	0.001	0.006	0.006	0.008	0.002

Although Armstrong et al. set their thresholds for carryover, change, and novelty borders in their quadratic models to [0, 25%], [26%, 75%], and [76%, 100%], we avoided such strict cutoffs as our data differed. Instead, we chose to map those concepts to the left, center, and right of the polynomial curves to have an approximate idea about the relationships. This mapping still followed the same order of concepts on the $novelty$ axis although the base of the quadratic curve was not always positioned on the 50%.

Our findings in Figures 6 and 7 show a convex behavior, indicating that C/C++ and OO C++ have high concept knowledge scores for carryover concepts. Namely, students who find these concepts familiar or less novel tend to score higher in related questions. While changed concepts seem to cause some decrease in the knowledge score, novel-perceived concepts tend to push the curve moderately higher. However,

right side of the curve cannot achieve the level that carryover concepts were able to reach. In conclusion, we observe that we have a partial curvilinear relationship for the students who transitioned to these two languages. While linear regressions cannot capture the slight increase for novel concepts, they do indicate that scores go down as novelty increases.

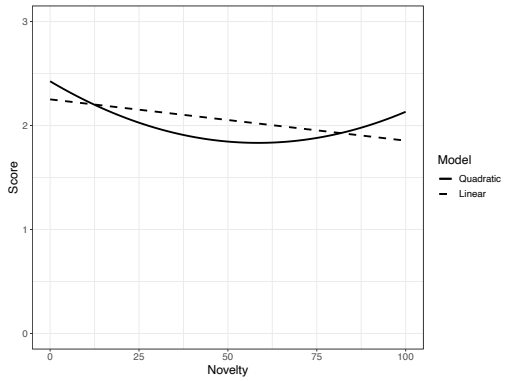

Fig. 6: Regression curves for C/C++

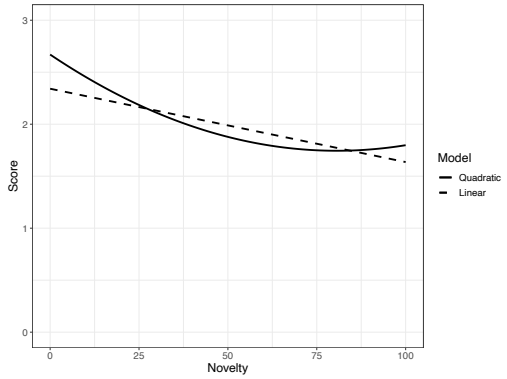

Fig. 7: Regression curves for OO C++

When it came to OO Java, however, we observed an entirely different behavior. Namely, both linear and quadratic models manifested an opposite behavior when compared to previous two cases. As seen in Figure 8, we no longer had an up-facing curve, meaning carryover concepts were associated with low knowledge scores, changed concepts led to higher scores, and the knowledge scores of novel concepts were not as low as those of carryover concepts. Moreover, linear regression line suggested increasing scores for higher levels of novelty perception in general with some exceptions after 75% novelty.

Since OO participants took the respective questionnaires with the same concepts and knowledge questions, we also combined these two groups to analyze the overall linear and quadratic trends. This merge led to a bell-shaped curve in Figure 9 which appeared to be influenced by OO Java observations. The slightly positive slope in the linear model suggests a very weak relationship between novelty perception and the scores, which can also be inferred from Table IV. By looking at the R^2 and p-values of both models, we observed that

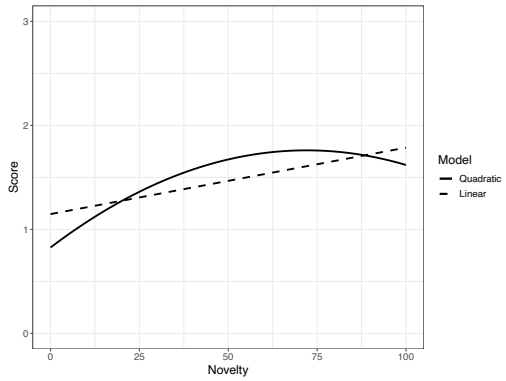

Fig. 8: Regression curves for OO Java

neither model adequately explained the variance nor identified novelty as a strong predictor.

TABLE IV: Fitted model features for combined datasets.

	OO Combined		**All combined**	
	Linear	**Poly**	**Linear**	**Poly**
α	1.708	1.735	2.029	1.878
β_1	0.0004	0.270	-0.003	-2.448
β_2	-	-1.644	-	1.427
R^2	0.0002	0.006	0.009	0.012
$p-value$	0.802	0.300	0.019	0.025

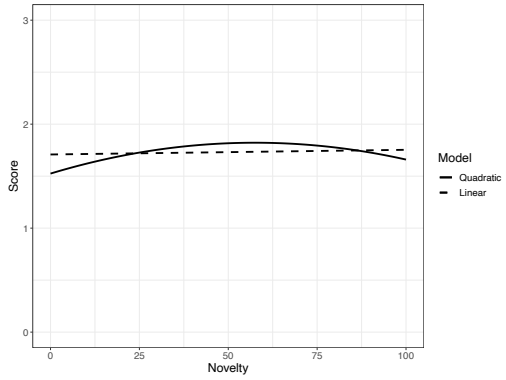

Fig. 9: Regression curves for OO C++ and OO Java combined

Finally, we performed the same analyses on 630 observations by combining all groups together despite their concept/topic differences. This new data set gave us a similar outcome to both C/C++ and OO C++ groups as shown in Figure 10, but still got affected by OO Java observations. Hence we obtained a wider curve which was incomplete on the right hand side and showing lower scores on that end when compared to Figures 6 and 7.

Upon examining R^2 values, we conclude that these values for the **All combined** model are not as good as the values for the **C/C++** and **OO C++** models. Even then, the **All combined model** has the capability to explain the variance better than the **OO combined** model. Similarly, the p values show that novelty had a stronger relationship with score for the **All combined** model than the **OO combined** model.

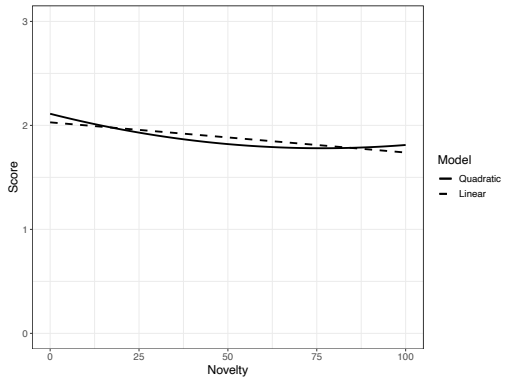

Fig. 10: Regression curves for all transitions combined

C. Comparative analysis of OO courses

Shown in section IV-B above the OO Java curve behaved much differently than that of the OO C++ curve. This notable discrepancy warranted further research, prompting us to probe deeper into both OO groups for a more detailed analysis.

In addition to Table II which statistically summarizes overall differences between these two cohorts, Figure 11 pictorially highlights a higher median score for OO C++ group. In fact, having the median value equal to its first quartile underlines a considerably lower performance among the OO Java participants. When compared to C/C++, OO Java appears to have a distribution which has scores accumulated in the lower end of the scores while OO C++ has a more similar density curve.

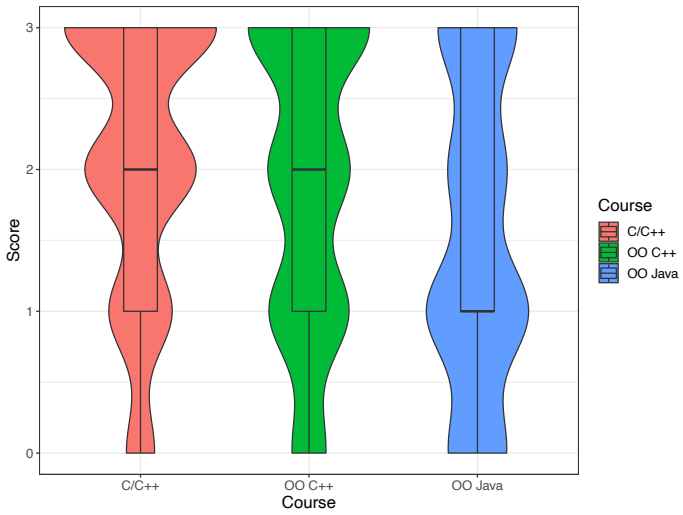

Fig. 11: Distribution of degree of knowledge score

While the box plots in Figure 11 do not show much difference in variance which may be partly due to limited range of the discrete scoring metric, Figure 12 suggests higher variation of perceived novelty in OO Java data ($\sigma = 33.62$) compared to OO C++ observations ($\sigma = 28.26$). Nevertheless both cohorts' novelty perception is still higher than that of C/C++ group which also has a higher variation ($\sigma = 34.14$). It is also evident from the density curves that OO C++ has more accumulation around 60% and 25% novelty, whereas accumulations are more towards the higher end for OO Java. C/C++ density distinguishes itself with almost a unimodal distribution which is right skewed, meaning novelty perceptions are mostly low.

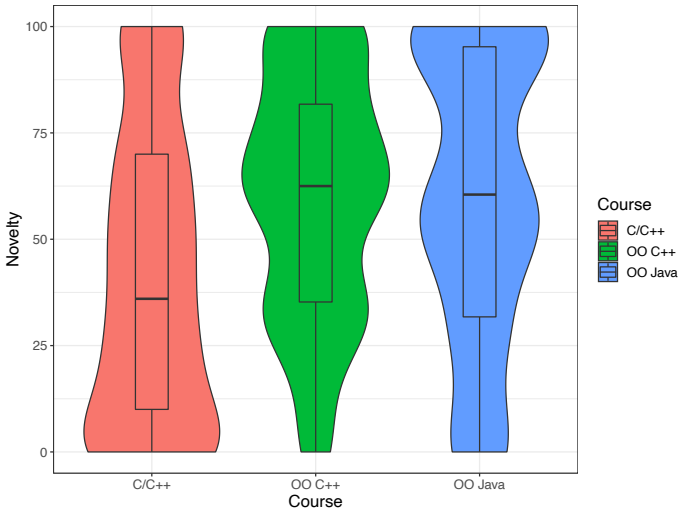

Fig. 12: Distribution of degree of perceived novelty

V. Discussion

Transitioning experience from one language to another is considered as an example of transfer of knowledge that can be further tied to Mindshift Learning Theory (MLT) which offers us the use of carryover, change, and novel concepts. Previous work, indeed, leverages such categorization when studying the challenges for those who have had programming exposure but not transitioning to a new language. While these transitions can take place at a professional level such as for software developers, they can also be possible for learners who follow a prerequisite chain in a college setting. Inspired by earlier research done on professional developers [21], we attempt to investigate the learning experiences of transitioning students from their perspectives and measure the concordance between their perceptions and actual knowledge.

Preliminary analysis showed that both quadratic and linear models mostly found perceived novelty as a significant factor for knowledge score except when we combined both OO C++ and OO Java observations. With this exception noted, we observed that curvilinear relationships were present between different levels of novelty perception and concept knowledge scores, aligning with findings from previous research [21]. However, our findings diverged in a way that quadratic models did not produce a full parabola shape and linear models manifested a subtle downward trend. Moreover, OO Java observations demonstrated an opposite trend and suggested a different curvilinear relationship for the perception levels.

One of the implications of contrasting outcomes of the models was that the student cohorts did not perform as well as the professionals from the original study on concepts they had

perceived as novel. Still they scored higher in novel concepts than changed concepts. When examining the same relationship in the OO Java group, we noticed further discordance which suggested that changed concepts led to higher knowledge scores.

The discrepancy in OO Java and partial concordance in other cohorts warranted further analysis which concluded that OO Java students had lower concept knowledge score in general with a high variation of novelty perception. Although C/C++ observations had a similar variation, students did not appear to find many concepts novel. This could be attributed to the concept differences for those transitioning from Python to C/C++ and C/C++ to OO Java. It is also worth noting that the degree of perceived novelty for a given concept was the only independent variable and it fell short in capturing the variance in concept knowledge score despite its statistical significance. Low R^2 values indicated that there might have been several other factors besides the perceived novelty that could impact the knowledge score.

There are several hidden factors that could contribute to the concept knowledge score and account for the divergence from the expected curvilinear relationship we wanted to compare our data against. Considering the student profile who completed the survey, we believe that timing of the courses taken may have impact on participants' concept knowledge score as students progress through the curriculum at different paces. Of the same token, it is unknown how often they used the concepts since their completion of the course(s). For instance, CS2 (Data Structures) is required by students in the Computer Science (CSC) program, and not required in the Computer Information System (CIS) program. In the latter group, there might be students who have a lower degree of perceived novelty for the concepts, but they might have answered the questions incorrectly due to lack of practice and familiarity.

Our respondents are also novice programmers who have diverse backgrounds from nine different majors with varying levels of experience in programming, whereas participants in [21] are professional software developers who had been programming on a daily basis as a key part of their job. Our cohort may even have students who are not actively programming in their current courses. Thus, our students likely have not mastered the concepts yet or an underlying understanding of computer programming principles, resulting in lower scores for novel concepts. Alternatively, some students may have utilized additional learning opportunities such as online learning platforms or personal programming related projects after completing the course.

Finally, our study attempts a multi-granular examination of transitioning not only between different languages, but also between different levels of the same language, such as from procedural C++ to OO C++. Therefore, we introduced two different sets of concepts which might have eventually caused some discrepancies in the aggregated analysis, where our intention was to perform a language-agnostic analysis for transitioning experience.

VI. Conclusions and Future Work

Transfer of Knowledge is crucial and inevitable in today's world, especially for a computing professional who is typically expected to program in several languages [48]. Students pursuing a computing program are also exposed to multiple programming languages in their curriculum. Our project investigates the cognitive load on students when they transition between multiple programming languages, especially due to changed concepts. An excessive cognitive load can result in low motivation for students, eventually leading to decreased retention.

In this paper, we report our findings from our initial exploration of how student knowledge is impacted by carryover, changed, and novel concepts when they transfer knowledge from one programming language to another. Our work in this paper was inspired by [21] which investigated a similar question when professional software developers transitioned from traditional programming to OO development. We studied the transition among three different courses in our curriculum: C/C++, OO C++, and OO Java. Our analysis showed that students in C/C++ and OO C++ tended to have a lower knowledge score as novelty increased. We noticed that students' exhibited lower knowledge for changed concepts as compared to more novel concepts, a trend similar to that in [21]. For OO Java, we found students report higher levels of novelty, and exhibit lower concept knowledge compared to C/C++ and OO C++. For OO Java, we found a trend different from C/C++ and OO C++ where students tended to have a higher knowledge score as novelty increased, and exhibit the highest knowledge for changed concepts.

As part of our future work, we plan to enrich our study by considering additional factors such as how students' concept knowledge changes an extended period after they had taken a course, and based on their use of concepts in other contexts following the course. Moreover, we plan to extend our study to other languages as well, such as SQL, and JavaScript. We also have interest to explore external factors to analyze our data in terms of how student demographics such as age, gender, and race influence transfer of knowledge in programming languages. Similarly, we want to explore the effect of a student's socioeconomic factors on their ability to transfer knowledge successfully. Our work presented in this paper, together with our planned future work, will go a long way towards the development of a comprehensive knowledge base regarding ToK in the computing curriculum, and how it impacts learning experience and student retention.

References

[1] (2012, January 25) Why We Need a 4th R: Reading, wRiting, aRithmetic, algoRithms . [Online]. Available: $http://dmlcentral.net/blog/cathy-davidson/why-we-need-4th-r-reading-writing-arithmetic-algorithms$

[2] S. Uludag, Karakus, Murat, and S. W. Turner, "Implementing IT0/CS0 with scratch, app inventor forandroid, and lego mindstorms," in *Proc. of the 2011 Conf. on Info. Tech. Edu (SIGITE)*, ser. SIGITE '11. New York, New York, USA: ACM Press, Oct 2011, p. 183.

[3] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[4] A. Perlis, *The computer in the university. In M. Greenberger, Ed., Computers and the World of the Future*. Cambridge, MA, USA: MIT Press, 1962.

[5] "Bureau of Labor Statistics, U.S. Department of Labor, Occupational Outlook Handbook, Computer and Information Technology Occupations," *accessed on May 11, 2024*, Last Modified Date: Wednesday, April 17, 2024. [Online]. Available: https://www.bls.gov/ooh/computer-and-information-technology/home.htm

[6] N. A. of Sciences, D. on Engineering, P. Sciences, C. Science, T. Board, Policy, G. Affairs, B. on Higher Education, and C. on the Growth of Computer Science Undergraduate Enrollments, *Assessing and responding to the growth of computer science undergraduate enrollments*. National Academies Press, 2018. [Online]. Available: https://nap.nationalacademies.org/read/24926/chapter/1

[7] T. Camp, W. R. Adrion, B. Bizot, S. Davidson, M. Hall, S. Hambrusch, E. Walker, and S. Zweben, "Generation cs: the growth of computer science," *ACM Inroads*, vol. 8, no. 2, pp. 44–50, 2017. [Online]. Available: https://cra.org/data/generation-cs/

[8] Berg, B., Holsapple, M., Ibrahim, M., Randolph, B., and Shapiro, D., "Undergraduate Degree Earners: Academic Year 2022-23," National Student Clearinghouse Research Center,Herndon, VA, April 2024. [Online]. Available: https://nscresearchcenter.org/undergraduate-degree-earners/

[9] J. L. Tims, C. Tucker, M. A. Weiss, and S. Zweben, "Computing enrollment and retention: Results from the 2021–22 undergraduate enrollment cohort," *ACM Inroads*, vol. 14, no. 4, pp. 24–43, 2023. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3629980

[10] Berg, B., Lee, S., Randolph, B., Ryu, M., and Shapiro, D., "Current Term Enrollment Estimates: Spring 2023," National Student Clearinghouse Research Center, Herndon, VA, 2023. [Online]. Available: https://nscresearchcenter.org/wp-content/uploads/CTEE_Report_Spring_2023.pdf

[11] C. Stephenson, A. D. Miller, C. Alvarado, L. Barker, V. Barr, T. Camp, C. Frieze, C. Lewis, E. C. Mindell, L. Limbird *et al.*, *Retention in computer science undergraduate programs in the us: Data challenges and promising interventions*. ACM, 2018.

[12] A. Luxton-Reilly, Simon, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard *et al.*, "Introductory programming: a systematic literature review," in *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education*, 2018, pp. 55–106.

[13] Karakus, Murat, S. Uludag, Guler, Evrim, S. W. Turner, and A. Ugur, "Teaching computing and programming fundamentals via App Inventor for Android," in *Int'l Conf. on Info. Tech. Based Higher Education and Training (ITHET)*. IEEE, Jun 2012, pp. 1–8.

[14] H. Bisgin, M. Mani, and S. Uludag, "Delineating factors that influence student performance in a data structures course," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–9.

[15] G. Alexandron, M. Armoni, M. Gordon, and D. Harel, "The effect of previous programming experience on the learning of scenario-based programming," in *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, 2012, pp. 151–159.

[16] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: students' perceptions of blocks-based programming," in *Proceedings of the 14th international conference on interaction design and children*, 2015, pp. 199–208.

[17] L. Goosen, "A brief history of choosing first programming languages," in *History of Computing and Education 3 (HCE3) IFIP 20 th World Computer Congress, Proceedings of the Third IFIP Conference on the History of Computing and Education WG 9.7/TC9, History of Computing*. Springer, 2008, pp. 167–170.

[18] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer science education*, vol. 13, no. 2, pp. 137–172, 2003.

[19] A. Dingle and C. Zander, "Assessing the ripple effect of cs1 language choice," in *Proceedings of the fourteenth annual consortium on Small Colleges Southeastern conference*, 2000, pp. 85–93.

[20] J. Gal-Ezer and D. Harel, "What (else) should cs educators know?" *Communications of the ACM*, vol. 41, no. 9, pp. 77–84, 1998.

[21] D. J. Armstrong and B. C. Hardgrave, "Understanding mindshift learning: the transition to object-oriented development," *MIS quarterly*, pp. 453–474, 2007.

[22] T. Galoyan and K. Betts, "Integrative transfer of learning model and implications for higher education," *The Journal of Continuing Higher Education*, vol. 69, no. 3, pp. 169–191, 2021.

[23] R. S. Woodworth and E. L. Thorndike, "The influence of improvement in one mental function upon the efficiency of other functions.(i)." *Psychological review*, vol. 8, no. 3, p. 247, 1901.

[24] E. L. Thorndike and R. S. Woodworth, "The influence of improvement in one mental function upon the efficiency of other functions. ii. the estimation of magnitudes." *Psychological Review*, vol. 8, no. 4, p. 384, 1901.

[25] M. K. Singley and J. R. Anderson, *The transfer of cognitive skill*. Harvard University Press, 1989, no. 9.

[26] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive science*, vol. 7, no. 2, pp. 155–170, 1983.

[27] T. A. Blaxton, "Investigating dissociations among memory measures: Support for a transfer-appropriate processing framework." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 15, no. 4, p. 657, 1989.

[28] J. Lave and E. Wenger, *Situated learning: Legitimate peripheral participation*. Cambridge university press, 1991.

[29] K. Beach, "Chapter 4: Consequential transitions: A sociocultural expedition beyond transfer in education," *Review of research in education*, vol. 24, no. 1, pp. 101–139, 1999.

[30] D. K. Detterman and R. J. Sternberg, *Transfer on trial: Intelligence, cognition, and instruction.* Ablex Publishing, 1993.

[31] D. N. Perkins and G. Salomon, "Knowledge to go: A motivational and dispositional view of transfer," *Educational Psychologist*, vol. 47, no. 3, pp. 248–258, 2012.

[32] G. Salomon and D. N. Perkins, "Rocky roads to transfer: Rethinking mechanism of a neglected phenomenon," *Educational psychologist*, vol. 24, no. 2, pp. 113–142, 1989.

[33] S. Leberman and L. McDonald, *The transfer of learning: Participants' perspectives of adult education and training*. Routledge, 2016.

[34] J. M. Royer, "Theories of the transfer of learning," *Educational psychologist*, vol. 14, no. 1, pp. 53–69, 1979.

[35] R. M. Gagne, "Contributions of learning to human development." *Psychological review*, vol. 75, no. 3, p. 177, 1968.

[36] J. M. Royer, "Introduction: Framing the transfer problem," *Transfer of learning from a modern multidisciplinary perspective*, pp. vii–xxvi, 2005.

[37] R. E. Haskell, *Transfer of learning: Cognition and instruction*. Elsevier, 2000.

[38] S. M. Barnett and S. J. Ceci, "When and where do we apply what we learn?: A taxonomy for far transfer." *Psychological bulletin*, vol. 128, no. 4, p. 612, 2002.

[39] R. Grossman and E. Salas, "The transfer of training: what really matters," *International journal of training and development*, vol. 15, no. 2, pp. 103–120, 2011.

[40] N. S. Rebello, D. A. Zollman, A. R. Allbaugh, P. V. Engelhardt, K. E. Gray, Z. Hrepic, and S. F. Itza-Ortiz, "Dynamic transfer: A perspective from physics education research," *Transfer of Learning: Research and Perspectives. Greenwich: Information Age Publishing*, 2004.

[41] J. D. Bransford and D. L. Schwartz, "Chapter 3: Rethinking transfer: A simple proposal with multiple implications," *Review of research in education*, vol. 24, no. 1, pp. 61–100, 1999.

[42] J. Lobato, "How design experiments can inform a rethinking of transfer andviceversa," *Educational researcher*, vol. 32, no. 1, pp. 17–20, 2003.

[43] S. R. Daffron and M. W. North, *Successful transfer of learning*. Krieger Publishing Company Malabar, FL, 2011.

[44] S. J. Gould and N. Eldredge, "Punctuated equilibria: the tempo and mode of evolution reconsidered," *Paleobiology*, vol. 3, no. 2, pp. 115–151, 1977.

[45] J. M. Bartunek and M. K. Moch, "First-order, second-order, and third-order change and organization development interventions: A cognitive approach," *The Journal of applied behavioral science*, vol. 23, no. 4, pp. 483–500, 1987.

[46] D. C. Gash and W. J. Orlikowski, "Changing frames: Towards an understanding of information technology and organizational change." in *Academy of Management Proceedings*, vol. 1991, no. 1. Academy of Management Briarcliff Manor, NY 10510, 1991, pp. 189–193.

[47] M. R. Louis and R. I. Sutton, "Switching cognitive gears: From habits of mind to active thinking," *Human relations*, vol. 44, no. 1, pp. 55–76, 1991.

[48] L. A. Meyerovich and A. S. Rabkin, "Empirical analysis of programming language adoption," *SIGPLAN Not.*, vol. 48, no. 10, p. 1–18, oct 2013.