Tools, Languages, and Environments Used in Primary and Secondary Computing Education

Monica M. McGill
Knox College
CSEdResearch.org
Galesburg, IL, USA
mmmcgill@knox.edu

Adrienne Decker
University at Buffalo
Buffalo, NY, USA
adrienne@buffalo.edu

ABSTRACT

With the advent of teaching primary and secondary computing education, tools, languages, and environments (TLEs) are important pedagogical support systems for students and teachers. While there are a number of resources available for teaching K-12 students and teachers, there is little synthesis of the data with respect to usage and adoption rates for various TLEs. Using data extracted from 510 articles related to K-12 education, we conducted an analysis using descriptive statistics to determine what TLEs in K-12 are most frequently studied by researchers. We found 193 TLEs being used in research studies and experience reports, then differentiate between these two types of data and between students and teacher professional development. This preliminary research provides a first descriptive analysis of TLEs being used in K-12 space and simultaneously sets the stage for creating a classification system for TLEs based on the literature, including how they are used and what topics (in or outside of computing education) they are used to teach.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Computing education programs**; **Computer science education**.

KEYWORDS

Languages, tools, environments, primary education, secondary education, K-12 education, research, experience reports

ACM Reference Format:
Monica M. McGill and Adrienne Decker. 2020. Tools, Languages, and Environments Used in Primary and Secondary Computing Education. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway*. ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/3341525.3387365

1 INTRODUCTION

With the increased emphasis on teaching computing in primary and secondary education across the globe [14, 27], there is also an increased interest in the tools, languages, and environments (TLEs) used to teach K-12[1] students and teachers [2, 7, 19]. These pedagogical support systems are not unlike many other manipulatives and resources used throughout education and are recognized by organizations as a critical part of learning and understanding computer science [2, 7].

TLEs at the post-secondary level have been of great interest to the computing education research community, though prior to 2007, interest in computing education at the primary and secondary levels did not appear frequently in documented discussions (via the ACM SIGCSE-Members listserv) [4]. Further, there are only limited references in the literature to the broader context of TLEs at the primary and secondary levels and no full synthesis or empirical investigation of the TLEs used.

Kelleher and Pausch (2005) understood the importance of TLEs in helping students tackle the many challenges they face when learning computing [17]. They developed a taxonomy for programming environments and languages for novice programmers in the context of the programming constructs the TLEs support and their "approaches to making programming more accessible to novice programers", further emphasizing the need of this research to remove both mechanical and sociological barriers to programming [17, p. 84]. Gomez-Albarran (2005) also developed a high-level domain structure for investigating 20 important tools [13]. More recently, Malmi, Utting, and Ko (2019) emphasized the importance of studying and evaluating TLEs in the context of student populations and learning environments as well as the use of learning analytics in these systems being considered at scale [19].

Based on this gap in the literature at the primary and secondary levels and the critical nature of TLEs in teaching computing to K-12 teachers and students, we considered several questions that would serve as a first step in a more comprehensive research agenda–including what would be of immediate interest to the community as we prepare to create a classification system for these tools to identify gaps. Our overarching research questions for this particular study became: *What tools, languages, and environments used to teach computing to primary and secondary students and teachers are most frequently studied by researchers and reported on in practice via experience reports?*

This study is important for informing the computing education community where research resources and efforts are most focused and where the field currently stands. This study provides TLE developers with a reference point in viewing the context in which TLEs

[1]While not all global primary and secondary education systems consist of 13 grades named K, 1, 2, ..12, "K-12" is becoming a widely used way of representing these grades. It is in this context that K-12 is used throughout this paper.

are referenced in the K-12 literature. More importantly, it provides a better understanding of TLEs so broader questions about how they are used, where they are used, and by whom can be considered.

2 BACKGROUND

2.1 Primary and Secondary Schools

There was little to be found in terms of systematic analysis of the TLE landscape in the schooling prior to university. Grover and Pea [15] published a comprehensive review of computational thinking (CT) in K-12 in 2013. They indicate that there are several programming tools available for supporting CT (e.g., robotics kits, Scratch, Alice, GameMaker, Agentsheets, and Arduino). However, their article does not address the prevalence of each of these tools in the landscape of published work in the area.

Outside of formal literature, several organizations provide resources for teachers to use in their classrooms. While none report adoption rates, we infer that their availability means that they are being used in the classroom. The Computer Science Teachers Association's (CSTA) is an organization that supports K-12 computing education in the United States and Canada. Their published standards for K-12 computing education [2] define learning objectives and achievements for students learning computing concepts in the K-12 space. Within the standards, no TLEs are specified, however, a listing of professional development opportunities for teachers [3] mentions some specific TLEs (Scratch, game-based programming).

The College Board provides Advanced Placement (AP) curriculum for two computing courses at the high school level primarily in the United States. For the AP Computer Science Principles course, the College Board provides a list of approved curricular providers[5]. This list mentions JavaScript (4 times), Python (4), Scratch (3), Processing (2), MIT AppInventor (1), PHP (1), Snap (1), SQL (1), and Swift(1). The College Board does not provide information about the percentage of usage of their suggested providers. The second course, AP Computer Science A, is described as an introductory college-level computer science course and expects Java to be used as the programming language [6].

CSforAll originated as a New York City initiative in 2010 and has evolved into providing resources for communities of providers, schools, funders, researchers, and teachers interested in providing computing education in K-12 schools [1]. Members register to be part of the community and provide information about what resources, TLEs or curriculum they have available. There are currently 268 content providers registered with the site. Likewise, Code.org arrived on the scene in 2013 with a mission to bring computing to a wider audience with the Hour of Code [8]. They have expanded their resources for supporting teachers to include curriculum for elementary, middle, and high school and provide an online work space for teachers to use with their students.

2.2 Post-Secondary Schools

Detailed information about TLE usage can be found in the ITiCSE working group report *Introductory Programming: A Systematic Literature Review*, which describes a literature review for introductory programming[18]. Examining data from years 2003 to 2017, the group examined 2,189 articles related to the teaching of introductory programming. This review offers a listing of programming environments and editors (section 6.4.2) as well as a listing of tools developed specifically for learning programming (section 6.4.4). Specific languages are mentioned, but statistics on how prevalent these languages are in the papers analyzed are not provided.

De Raadt, Watson and Toleman (2002) created a census of what all Australian universities were teaching in their introductory programming courses in 2001 [10]. They reported on specific programming language usage as well as specific IDE usage. This census was repeated in 2003 [11], 2010 [22], 2013 [21], and 2016 [23]. Mason and Simon (2017) present longitudinal data about the trends over the 15 years to show the rise and fall of popularity of languages and IDEs during that time [23].

In 2011, Davies, Polack-Wahl, and Anewalt surveyed 371 colleges and universities within the United States and identified programming languages used in CS0 and CS1 courses, reporting that 47% of CS1 classes use a Graphical IDE, 15% use a command-line IDE, and 37% use a mix of both [9]. Likewise, Guo [16] looked at languages used in CS0 and CS1 of the top 39 ranked US computer science departments and found that the only languages used were Python, Java, Matlab, C, C++, Scheme, and Scratch. The article does not differentiate between which languages were used in CS0 or CS1.

Murphy, Crick, and Davenport give a report on Introductory Programming Courses in the UK in 2017 [26]. Their report includes information from 80 instructors from at least 70 institutions. They report on specific languages and IDEs used in the courses. This study uses the same questions and answer options as the earlier study from Australasia [21].

2.3 Summary

Formal studies on the landscape of TLEs have been conducted at the post-secondary level, though these studies are limited and only one attempt at a taxonomy has been conducted. While it is clear that there are a number of resources and resource centers available for teachers interested in teaching students prior to post-secondary, aside from the work in computational thinking [15], none of the data is synthesized with usage/adoption rates for the various TLEs, nor have these TLEs been aligned with curriculum standards. Further, there is no method of determining what TLEs have the most impact on academic achievement for the various standards.

3 METHODOLOGY

Malmi, Utting, and Ko (2019) [19] provide a categorical definition of tools, including software applications, web applications providing some service, software frameworks, and definition languages used to teach computing. Their categorization excludes professional tools, physical computing systems, and unplugged activities. However, in published articles, a broad range of tools used in education are referenced that may not have been specifically designed for teaching computing [19]. For example, languages like Java and JavaScript were not created to teach programming, but nevertheless are used widely in computing education.

In many cases, it is easy to distinguish between tools, programming languages, and development environments. C, for example, is easy to identify as a language. Brackets, NotePad++, Sublime or other editors are tools for editing code. An environment for developing code may include the editors, the terminal window, and the compiler that is used to compile (and execute) the code. In other cases, an editor or integrated development environment (IDE) may be created specifically for the purpose of one language and the IDE cannot be separated from it. For example, Scratch could be classified as an IDE but the block-styled language that is used to program within Scratch cannot be separated from the IDE [20].

In some forms of teaching computing, particularly computational thinking, physical manipulatives are used. Though these materials are somewhat abstract, they have been shown to be effective in teaching concepts to a variety of age groups. We refer to these as tools, rather than materials, since materials can convey medium like textbooks, forums, and journal entries. Likewise, the process of capturing unplugged activities and products designed for primary and secondary education requires a broader definition of tools, languages, and environments than defined by Malmi, et. al., or than currently exist in the field of professional software development.

Given the current lack of formal definitions that encompass all aids mentioned in the literature, we use the term tools, languages, and environments to broadly capture those formal materials that have been studied in research studies or mentioned in experience reports as part of the overall student experience. This is a necessary first-step in understanding the many types of TLEs used so that a detailed ontology can be developed in the future. The following sections describe the data source from which this study derives and the analysis conducted on the data.

3.1 Data Source

The data used for this study is a subset of the publicly available dataset from CSEdResearch.org, a site that houses summaries of articles related to primary and secondary computing education [12, 24, 25]. The initial set of 510 articles focused on primary and secondary computing education across ten publication venues (2012-2018) were included in this analysis. The dataset represents manually curated data from the articles and the articles originated from ten publication venues, including ACM and IEEE journals and conference proceedings related to computing education, Computer Science Education, and Koli Calling[2] [24]. To identify these articles, all of the abstracts for all of these venues were carefully read to determine if they were focused on primary and secondary education. If the researchers determined that it was, then the researchers either manually curated the data or gave the articles to the data curation team (undergraduate students [25]). If the data was curated from the undergraduate student team, then the researchers provided a second review to validate the data.

Though many of the publication venues in which the articles appear and the data is extracted are open to all countries, the reporting of the TLEs comes primarily from the United States (U.S.), with 185 (50.1%) of the papers covering students located in the U.S., followed by the United Kingdom with 19 (5.1%), Brazil with 16 (4.3%), Israel with 14 (3.8%), and Spain with 11 (3.0%).

[2]ACM International Computing Education Research, ACM Innovation and Technology in Computer Science Education, ACM SIGCSE Technical Symposium on Computer Science Education, ACM Transactions on Computing Education, Frontiers in Education, IEEE Global Engineering Education Conference, IEEE Transactions on Education, Journal of Educational Computing Research, Koli Calling, Taylor & Francis' Computer Science Education

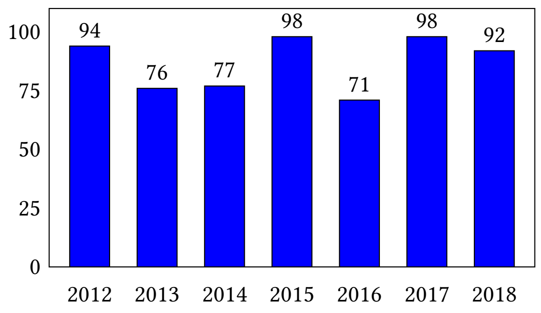

Figure 1: Number of TLEs mentioned in studies/reports across the years

3.2 Data Analysis

The CSEdResearch.org dataset is stored in a MySQL database. To extract data related to this study, one of the researchers constructed and executed over a dozen SQL queries over the entire set of data, extracting the pre-specified subsets of the data to be analyzed. The results were verified and then exported and stored into comma-separated value (.csv) files. Only descriptive statistics (count and percentage) were calculated for the purposes of this study. The descriptive statistics were calculated for the following data:

- TLEs used in informal and formal learning environments across the entire dataset
- TLEs used in professional development (PD) studies
- TLEs used in studies of students (total), those used in research studies of students, and those used in experience reports of teaching students
- TLEs used across the countries where students were located

To contextualize the impact and usage of these tools, separate SQL queries were created and executed so that we could provide the number of participants in these studies and experience reports at the aggregated level. Though we understand the results can provide context, these numbers reflect an entire study and may be over-inflated in some instances due to the type of study being conducted and the manner the data was presented in the original article. For example, if the study was a treatment-control group study, and 50 students participated in the study with 25 in each group, only 25 students might have been exposed to the TLE.

4 RESULTS

Of the articles in the original dataset, 247 (48%) were research articles, 236 (46%) were experience reports, and 27 (5%) were position papers. For the purposes of this analysis, we exclude position papers in the study in order to focus on activities and curriculum described in the research articles and experience reports. We found 193 TLEs as part of an activity or curriculum reported within the research articles and experience reports. Figure 1 shows the number of TLEs appearing in articles over the years.

4.1 General Overview

To determine which TLEs are most often referenced, we first ran a query on the TLEs referenced throughout articles that are focused on both teachers (through professional development) and students.

Table 1: Most reported TLEs listed by article count (includes students and teachers as learners), N=605. Note that many studies include more than one TLE.

TLE	#	%
Scratch	85	14.0%
Java	25	4.1%
Python, AppInventor	24	4.0%
Alice, Arduino (non-Lilypad)	22	3.6%
CS Unplugged, Lego	21	3.5%
Lego Mindstorms	19	3.1%
Pololu 3PIs	12	2.0%
Greenfoot	11	1.8%
C#, HTML, Alice, Lego NXT, Processing	8	1.3%
AgentCubes, CSS, Snap!, Logo	7	1.2%

Table 2: Most reported TLEs listed by number of learners (students and teachers as learners) reported in articles, N=419.

TLE	#	%
Scratch	116,723	55.9%
AgentSheets	11,064	5.3%
AgentCubes	10,413	5.0%
Alice	7,393	3.5%
Java	6,751	3.2%
CSS	6,673	3.2%
Python	6,354	3.0%
Arduino (Not LilyPad)	6,163	3.0%
Bebras Challenge	6,081	2.9%
Greenfoot	6,001	2.9%

The most reported TLE was Scratch (14.0% of articles), followed by Java (4.1%), Python and AppInventor (4.0% each), and Alice and Arduino (non-Lilypad) (3.6% each) (see Table 1).

We provide a snapshot of the primary and secondary learner usage (students and teachers taking PD) of these TLEs in the 369 research studies and experience reports in Table 2, with the caveat described in the data analysis section above. The learner usage of these tools in their entirety is impressive, with a reach of 208,688 learners. Of these learners, approximately 116,723 (56.0%) used Scratch. The next closest TLEs are AgentSheets and AgentCubes with 11,064 (5.3%) and 10,413 (5.0%) learners, respectively, studied or reported.

4.2 Students as Learners

We analyzed the data by considering those articles that reported on research studies or experience reports with primary and secondary students as the participants. There were 208,071 student participants in the articles, with Scratch being studied and used in the majority (77 articles (14.4%)), followed by Java, AppInventor, Arduino (not LilyPad), Python, and Alice (see Table 3).

Analyzing the data based on the number of participants reported in the study or report, Scratch is again on top, with 116,434 (56.0%) students (see Table 4). AgentSheets and AgentCubes follow, garnering 5.3% and 5.0% respectively.

Table 3: Most reported TLEs listed by article count, N=535, primary and secondary students only.

TLE	#	%
Scratch	77	14.4%
Java	24	4.5%
AppInventor	21	3.9%
Arduino (not LilyPad), Python	20	3.7%
Alice, CS Unplugged	17	3.2%
Lego Mindstorms	15	2.8%
Greenfoot, Pololu 3PIs	10	1.9%
C#, Processing, HTML, Alice 2.2	8	1.5%
CSS, Logo, Lego NXT, AgentCubes, Snap!	7	1.3%
EarSketch, JavaScript, LilyPad Arduino	6	1.1%

Table 4: Most reported TLEs listed by number of primary and secondary students (N=208,071) in the study or report.

TLE	#	%
Scratch	116,434	56.0%
AgentSheets	11,057	5.3%
AgentCubes	10,406	5.0%
Alice	7,331	3.5%
Java	6,747	3.2%
CSS	6,673	3.2%
Python	6,332	3.0%
Arduino (Not LilyPad)	6,163	3.0%
Bebras Challenge	6,081	2.9%
Greenfoot	6,001	2.9%

4.2.1 Research Studies. We excluded experience reports and analyzed the research studies where primary and secondary students were participants. Within this subset of the data, Scratch is the most frequently researched, appearing in 45 (18.5%) of the studies, followed by AppInventor and CS Unplugged (4.9%), and Alice (4.1%) as shown in Table 5.

When analyzing the data by number of students as participants in the studies, Scratch again ranked number one with 13,953 participants (18.6%), followed by AgentCubes and AgentSheets (10,000 each at 13.3%), Alice (6,731 at 9.0%), and CSS (6,648 at 8.8%) (see Table 6).

4.2.2 Experience Reports. We analyzed the experience reports where primary and secondary students were the participants. Within this subset of the data, Scratch is the most frequently researched, appearing in 32 (11.0%) of the studies, followed by Java (5.9%), Python (4.5%), and Arduino (4.5%) as shown in Table 7.

When analyzing the data by number of students as participants in the studies, Scratch again rated highest with 102,580 participants (77.0%), followed by Bebras Challenge (4.6%), Python (4.5%), and Arduino and AppInventor (4.4% each) (see Table 8).

Table 5: Most reported TLEs listed by those used in research studies (N=243) most frequently (includes students and teachers as learners).

TLE	#	%
Scratch	45	18.5%
AppInventor, CS Unplugged	12	4.9%
Alice	10	4.1%
CS Unplugged, Lego Mindstorms, Python	7	2.9%
C#	5	2.1%
CSS, Logo, Snap!, Lego NXT, Kodu	4	1.6%

Table 6: Most reported TLEs listed by those researched most frequently (includes students and teachers as learners, N=75,196).

TLE	#	%
Scratch	13,953	18.6%
AgentCubes, AgentSheets	10,000	13.3%
Alice	6,731	9.0%
CSS	6,648	8.8%
Java	6,240	8.3%
Greenfoot	5,878	7.8%
CS Unplugged	1,720	2.3%
Bootstrap	1,674	2.2%
C#	1,337	1.8%
Jypeli	1,018	1.4%

Table 7: Most reported TLEs listed by those used most frequently in experience reports (N=290).

TLE	#	%
Scratch	32	11.0%
Java	17	5.9%
Python, Arduino (not LilyPad)	13	4.5%
AppInventor	9	3.1%
Lego Mindstorms	8	2.8%
Pololu 3PIs, Greenfoot, Alice	7	2.4%
Processing	6	2.1%
C++, CS Unplugged	5	1.7%
Alice 2.2, HTML, AgentCubes	4	1.4%

4.3 Professional Development

We examined the types of TLEs used in PD that are being studied (see Table 9). As expected, the list starts to reflect differences; however, Scratch is still the most frequently used TLE in PD.

Some articles were focused on teachers as learners in professional development, with the number of students (if known) that they teach provided within the articles or reports. Scratch still rated the highest, with 8 articles reporting that it was used as a TLE (see Table 10). Only the number of times the TLEs are reported in the articles focused on PD is presented here, since the data for the learner impact would need more in-depth analysis of the articles to ensure data integrity.

Table 8: Most reported TLEs listed by number of K-12 students (N=133,274) engaged in the experience and reported in experience reports.

TLE	#	%
Scratch	102,580	77.0%
Bebras Challenge	6,081	4.6%
Python	5,941	4.5%
Arduino (not LilyPad)	5,869	4.4%
AppInventor	1,126	4.4%
AgentSheets	1,057	0.8%
Scalable Game Design	894	0.7%
Alice 2.2	656	0.5%
Alice	600	0.5%
Java	507	0.4%
Pololu 3PIs	438	0.3%

Table 9: Most reported TLEs listed by number of teachers receiving PD, N=635.

TLE	#	%
Scratch	299	47.3%
CS Unplugged	71	11.2%
NetLogo	66	10.4%
Alice	62	9.8%
HTML	32	5.1%
Python	22	3.5%
Moodle	19	3.0%
Stencyl	16	2.5%
AgentCubes, AgentSheets, Scalable Game Design	7	1.1%
Lego Mindstorms, Lego NXT-G	5	0.8%

Table 10: Most reported TLEs listed by number of teachers taking PD that are reported in articles (N=75).

TLE	#	%
Scratch	8	10.8%
Alice	5	6.8%
CS Unplugged, Lego Mindstorms, Python	4	5.4%
AppInventor	3	4.1%
Java, Bebras Challenge, Arduino (not LilyPad), Alice, Lego NXT-G, Polulo 3PIs, Scratch Jr., Microsoft Kinect, Google Drive	2	2.7%

5 DISCUSSION

The results support previous studies, indicating that some (though not much) movement has been made in TLEs, and illustrate the new additions to the set commonly used in K-12 computing education. The results empirically confirm Grover and Pea's (2013) findings that Scratch, Alice, Kodu, Greenfoot, Arduino boards, AgentSheets and AgentCubes are popular TLEs [15]. Scratch is mentioned by the CSTA as well as game programming. Though we cannot confirm in this particular study whether or not TLEs are being used in the context of game programming (outside of PyGame, GameMaker, GameFroot, Scalable Game Design and XNA Game Studio, all of which appear as a TLEs), we can confirm Scratch is a well-vetted TLE that may offer a hearty learning experience for students [3].

Our results align with TLEs mentioned by the College Board, since JavaScript, Python, Scratch, Processing, AppInventor appear on their list [7]. PHP, Snap, SQL and Swift are all TLEs that do not appear as any top TLEs. Code.org also appears, although it hasn't been studied or mentioned in experience reports as frequently [8]. Though we did not analyze the content providers on the CSforAll site, this could be a potential way to find current information about TLEs that are used more casually–without formal studies or experience reports conducted using them [1].

These findings also align with the results of reviews of TLEs used for teaching computing at the post-secondary level. Java, Python, Scratch, and AppInventor all appear in both lists. C++ is referenced in only 5 papers total. However, ActionScript, Pascal, and Karel++ do not appear in our dataset since they are not reported in experiences or studies with K-12 students or teachers. Java, a popular choice among CS1 courses, appears infrequently in research articles. With the TLEs in our Background section focused on novice programmers, it makes sense that this overlap exists.

5.1 Limitations

One of the more pressing limitations is that the data curation process can only be as thorough as the data reported in the articles. Despite accepted best practices, not all authors report the number of participants in the study or experience. Therefore, the actual counts may be greater for some tools than others. Though there are continual efforts to improve reporting throughout the computing education community, we hope this study further illuminates the need for more accurate reporting in publications. In addition, the articles accepted to these publications may suffer from reviewer bias. Reviewers may be more inclined, for example, to accept or reject Scratch due to its popularity.

The data was manually curated from the articles. Despite the fact that each article underwent two reviews, data entry errors may have occurred. We also note that the dataset predominantly includes studies and reports conducted in the United States. This may skew the data and leaves room for further analysis of countries outside the U.S. for a comparison and contrast study.

Several articles may have been written on one study, increasing the likelihood that the TLEs mentioned are used with fewer participants and classrooms than they really are. We emphasize throughout the results and discussion section that what we are measuring is *the frequency in which they appear in studies*, not in the learning environments.

Occasionally, some articles include students who are undergraduates. For example, a camp may have been taught by undergraduate students to high school students or a particular study may have included both high school and undergraduate students. Though these appear infrequently in the database, it may have affected the participant numbers. This is in addition to the preemptive caveat mentioned in the Methodology section that we repeat here. Participant numbers may be over-inflated in some instances due to the type of study conducted and how data was presented in the original article. For example, if the study was a treatment-control group study, and 50 students participated in the study with 25 in each group, only 25 students might have been exposed to the TLE.

5.2 Future Research

The purpose of this study was to begin to understand the data and how it can be analyzed further. We are particularly interested in pursuing research to answer the following questions:

- *What would a current, comprehensive taxonomy/ontology for classifying TLEs look like as computing education expands into K-12?*
- *What has been the evolution of these tools, languages, and environments in primary and secondary education?*
- *What types of studies are still needed with respect to tools, languages, and environments to understand their efficacy in primary and secondary settings?*
- *How can we map existing TLEs against K-12 standards to find gaps based on the needs of primary and secondary computing education?*

In addition to those much larger picture research questions, additional data analysis could be conducted to answer other questions. For example, what are the changes of TLEs used over time? What are the differences in TLEs used in informal versus formal education? What are the differences within primary and secondary education–such as early primary, late primary and secondary education? This additional analysis may be of interest to researchers and educators as the K-12 computing education field takes shape.

6 CONCLUSION

TLEs are of unique interest to computing education researchers. Not only is it important to understand their impact on academic achievement, including social-behavioral factors like self-efficacy and sense of belonging, but it is also important to understand the context in which TLEs are used (e.g., topics, student demographics). These critical research areas can provide the data needed to improve these products that are oftentimes created by other computing education researchers. This research can also provide us with a greater understanding of what are the most effective TLEs to use for various demographic groups, provide data on topic areas that are well-covered by TLEs, and help identify gaps where additional enhancements to existing TLEs or new TLEs could be developed.

The lack of knowledge of what TLEs have been used with what correlating outcomes is limiting our current understanding of how to best deliver computing education in the K-12 space. Though we are limited in space with this study to investigate these more major questions, this study serves as another step towards addressing these questions on a larger scale.

7 ACKNOWLEDGEMENTS

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. 1625005, 1625335, 1757402, 1745199, and 1933671.

REFERENCES

[1] 2019. CSforAll. Retrieved June 20, 2019 from https://csforall.org

[2] 2019. CSTA CS Standards. Retrieved January 15, 2020 from https://csteachers.org/page/standards

[3] 2019. CSTA Professional Development Opportunities. Retrieved January 15, 2020 from https://csteachers.org/page/pdmatrix

[4] Austin Cory Bart and Clifford A. Shaffer. 2019. What Have We Talked About?. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. ACM, New York, NY, USA, 175–180. https://doi.org/10.1145/3287324.3287441

[5] College Board. 2019. Adopt Ready-To-Use Curricula. Retrieved June 20, 2019 from https://apcentral.collegeboard.org/courses/ap-computer-science-principles/classroom-resources/curricula-pedagogical-support

[6] College Board. 2019. AP 2018 Computer Science A: A Chief Reader Report. Retrieved June 20, 2019 from https://apcentral.collegeboard.org/pdf/ap18-computer-science-a-chief-reader-report.pdf?course=ap-computer-science-a

[7] College Board. 2019. AP Computer Science A: The Course. Retrieved June 20, 2019 from https://apcentral.collegeboard.org/courses/ap-computer-science-a/course?course=ap-computer-science-a

[8] Code.org. 2017. Code.org. Retrieved June 21, 2019 from https://code.org

[9] Stephen Davies, Jennifer A. Polack-Wahl, and Karen Anewalt. 2011. A Snapshot of Current Practices in Teaching the Introductory Programming Sequence. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 625–630. https://doi.org/10.1145/1953163.1953339

[10] Michael De Raadt, Richard Watson, and Mark Toleman. 2002. Language trends in introductory programming courses. In *Proceedings of the 2002 Informing Science+ Information Technology Education Joint Conference (InSITE 2002)*. Informing Science Institute, 229–337.

[11] Michael de Raadt, Richard Watson, and Mark Toleman. 2004. Introductory Programming: What's Happening Today and Will There Be Any Students to Teach Tomorrow?. In *Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30 (ACE '04)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 277–282. http://dl.acm.org/citation.cfm?id=979968.980005

[12] Adrienne Decker and Monica M. McGill. 2019. A Topical Review of Evaluation Instruments for Computing Education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* ACM.

[13] Mercedes Gómez-Albarrán. 2005. The teaching and learning of programming: a survey of supporting software tools. *Comput. J.* 48, 2 (2005), 130–144.

[14] Gov.uk. 2013. National curriculum in England: computing programmes of study. Retrieved August 29, 2017 from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

[15] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational researcher* 42, 1 (2013), 38–43.

[16] Philip Guo. 2014. Python is now the most popular introductory teaching language at top US universities.

[17] Caitlin Kelleher and Randy Pausch. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)* 37, 2 (2005), 83–137.

[18] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion)*. ACM, New York, NY, USA, 55–106. https://doi.org/10.1145/3293881.3295779

[19] Lauri Malmi, Ian Utting, and Andrew J Ko. 2019. Tools and Environments. In *The Cambridge Handbook of Computing Education Research*, Sally A Fincher and Anthony V Robins (Eds.). Cambridge University Press, Chapter 21, 639–662.

[20] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *Trans. Comput. Educ.* 10, 4, Article 16 (Nov. 2010), 15 pages. https://doi.org/10.1145/1868358.1868363

[21] Raina Mason and Graham Cooper. 2014. Introductory Programming Courses in Australia and New Zealand in 2013 - Trends and Reasons. In *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148 (ACE '14)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 139–147. http://dl.acm.org/citation.cfm?id=2667490.2667507

[22] Raina Mason, Graham Cooper, and Michael de Raadt. 2012. Trends in Introductory Programming Courses in Australian Universities: Languages, Environments and Pedagogy. In *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123 (ACE '12)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 33–42. http://dl.acm.org/citation.cfm?id=2483716.2483721

[23] Raina Mason and Simon. 2017. Introductory Programming Courses in Australasia in 2016. In *Proceedings of the Nineteenth Australasian Computing Education Conference (ACE '17)*. ACM, New York, NY, USA, 81–89. https://doi.org/10.1145/3013499.3013512

[24] Monica M. McGill and Adrienne Decker. 2019. Computer Science Education Resource Center. https://csedresearch.org

[25] Monica M McGill, Adrienne Decker, and Zachary Abbott. 2018. Improving Research and Experience Reports of Pre-College Computing Activities: A Gap Analysis. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education.* ACM, 964–969.

[26] Ellen Murphy, Tom Crick, and James H Davenport. 2016. An analysis of introductory programming courses at UK Universities. *arXiv preprint arXiv:1609.06622* (2016).

[27] Megan Smith. 2016. Computer Science for All. https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all