50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research

Brett A. Becker
University College Dublin
Ireland
brett.becker@ucd.ie

Keith Quille
Institute of Technology Tallaght
Ireland
keith.quille@it-tallaght.ie

ABSTRACT

The SIGCSE Technical Symposium is celebrating its 50th year, and a constant theme throughout this history has been to better understand how novices learn to program. In this paper, we present a perspective on the evolution of introductory programming education research at the Symposium over these 50 years. We also situate the Symposium's impact in the context of the wider literature on introductory programming research. Applying a systematic approach to collecting papers presented at the Symposium that focus on novice programming / CS1, we categorized hundreds of papers according to their main focus, revealing important introductory programming topics and their trends from 1970 to 2018. Some of these topics have faded from prominence and are less relevant today while others, including many topics focused on students—such as making learning programming more appropriate from gender, diversity, accessibility and inclusion standpoints—have garnered significant attention more recently. We present discussions on these trends and in doing so, we provide a checkpoint for introductory programming research. This may provide insights for future research on how we teach novices and how they learn to program.

CCS CONCEPTS

• **Social and professional topics → Computer science education; CS1**.

KEYWORDS

CS1; CS 1; CS-1; programming; introductory programming; introduction to programming; novice programming; survey; review

ACM Reference Format:
Brett A. Becker and Keith Quille. 2019. 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287432

1 INTRODUCTION

The SIGCSE Technical Symposium (the Symposium) is celebrating its 50th year, and researching how novices learn to program has been a consistent theme throughout this history. We present a perspective on this research over these first five decades. We also situate the Symposium and its impact in the context of the wider literature on introductory programming research. From an initial selection of 777 papers presented at the Symposium, we categorized 481 CS1 papers and offer a discussion on their topics and trends, providing a checkpoint for introductory programming research going forward into the second half of the Symposium's first century. This may prove insightful for future research on how we teach novices and how they learn to program. Our research goals are:

- RG1 Identify the important topics in introductory programming education research, including their trends, over the first 50 years of the SIGCSE Technical Symposium
- RG2 Situate the introductory programming research presented at the SIGCSE Technical Symposium in the context of the wider literature

2 RELATED WORK

Despite the large volume of work on introductory programming (commonly called 'CS1' [19]), there are relatively few surveys, reviews and meta-analyses with wide scope. Most relevant to the present work as it only considered papers presented at the Symposium, is a 2004 paper by Valentine [44], which focused on CS1 and CS2, including papers, workshops and panels. These were organized into a 'taxonomy' based on the paper approach (Marco Polo, Tools, Experimental, Nifty, Philosophy or John Henry), while our categorization is based on paper content/topic. A 2002 paper by Vasiga also provided data on CS1 and CS2 at SIGCSE, and stated: "To say there has been much discussion of CS1 and CS2 in SIGCSE circles is a gross understatement" [45, p. 28]. Table 1 shows the number of papers reported by these authors compared to the subsets of our papers that align with their dates. It is important to note that we only considered papers on CS1 (not CS2). We explain our method in Section 3.

Another effort relevant to our work is a 2018 ITiCSE Working group led by Luxton-Reilly & Simon which performed a large literature review of introductory programming research, citing over 700 references [29]. We discuss this further in Section 4.

	Valentine [44]	Vasiga [45]	this study
1984-1993	17	-	36
1994-2003	28	-	98
2000	-	18	8

Table 1: Number of papers reported at the Symposium on CS1 & CS2 [44, 45] and CS1 (this study). Note that these are subsets of our data as our study spans 1970-2018.

3 METHOD

We chose to analyze papers presented at the SIGCSE Technical Symposium focusing on introductory programming courses at the university level, commonly called CS1 [19]. We conducted a preliminary search of the ACM Digital Library (ACM DL) on January 21, 2018 and carried out a short feasibility study, mainly to refine the search parameters and method. On August 3, 2018 we conducted our final search, which was for papers including any of the following terms in the title, abstract, or keywords: CS1, "CS 1", "introductory programming", "introduction to programming", "novice programming", "novice programmers". This search returned 3,153 hits in the ACM DL Guide to Computing Literature, and 2,354 hits in the ACM DL Full-Text Collection. We then extracted all papers sponsored by SIGCSE (1,442 hits) and then filtered, leaving only papers presented at the SIGCSE Technical Symposium (777 hits). We then applied the exclusion criteria: (1) Papers less than three pages in length (eliminating 244), and (2) Papers not focusing on first-year university-level introductory programming (eliminating 52). This resulted in a final list of 481 papers. Our actual query was:

acmdlTitle: (CS1 "CS 1" "introductory programming" "introduction to programming" "novice programming" "novice programmers") OR recordAbstract: (CS1 "CS 1" "introductory programming" "introduction to programming" "novice programming" "novice programmers") OR keywords.author.keyword: (CS1 "CS 1" "introductory programming" "introduction to programming" "novice programming" "novice programmers")

3.1 Classification of papers

We acknowledge the inherent difficulty of reliably coding academic papers [41]. As this is not a systematic literature review, we allowed categories and subcategories to emerge as Author 1 processed all 481 papers. Our main goal was to categorize the papers fitting our criteria in a way that would accurately describe the landscape of CS1 research at the Symposium, and how it has changed in the last 50 years. For this reason we chose to categorize each paper into one category only (a many-to-one mapping). We discuss this and other threats to validity in Section 6. Once initial categories and subcategories were determined through Author 1 processing all 481 papers, Author 2 re-categorized a random sample of 10% placing 41 of 48 papers in the same categories, indicating 85% agreement. A third faculty member at the same institution as Author 1 repeated this re-categorization with a new random 10% sample and placed 40 of 48 papers in the same category as Author 1, indicating 83% agreement. After this, Authors 1 & 2 discussed the categories and made minor refinements resulting in eight top-level categories: (1) *First languages & paradigms*; (2) *CS1 design, structure & approach*; (3) *CS1 content*; (4) *Tools*; (5) *Collaborative approaches*; (6) *Teaching*; (7) *Learning & Assessment*; and (8) *Students*. In the interest of making this work useful for the community, details on these 481 papers including URLs, number of citations, and citations/year are included in a CSV file available at www.brettbecker.com/sigcse2019.

4 SITUATING THE SIGCSE TECHNICAL SYMPOSIUM IN THE WIDER LITERATURE

Before looking within the Symposium and the evolution of introductory programming education research over the last 50 years, we wanted to gain a picture of the Symposium within the broader literature. As mentioned in Section 2, Luxton-Reilly et al. conducted a 2018 review of introductory programming literature [29], using a similar but slightly broader search query and a much wider search space (not limited to the ACM Digital Library). This resulted in over 5,000 papers, reduced to 1,844 after applying exclusion criteria. We started with 1,442 papers which reduced to 481, approximately 26% of the 1,844 from the ITiCSE working group. This quick comparison reveals that the SIGCSE Technical Symposium represents not only a considerable space in the literature of introductory programming literature indexed by the ACM, but a healthy percentage of the introductory programming literature in general.

To gain a different perspective, we performed our search against: the ACM Digital Library Guide to Computing Literature; all papers published by ACM; and, all proceedings sponsored by SIGCSE. Table 2 shows our findings as a percentage of the hits from the Guide to Computing Literature. Like the comparison to the ITiCSE working group mentioned above, this shows that the SIGCSE Technical Symposium occupies a significant footprint in the wider literature; approximately 25% by this measure, almost the same as the 26% when compared to [29] in the above paragraph.

Search Space	hits	% total
ACM Digital Library Guide to Computing Literature	3,153	100%
Published by ACM	1,823	~58%
Conferences sponsored by SIGCSE	1,442	~46%
SIGCSE Technical Symposium	777	~25%

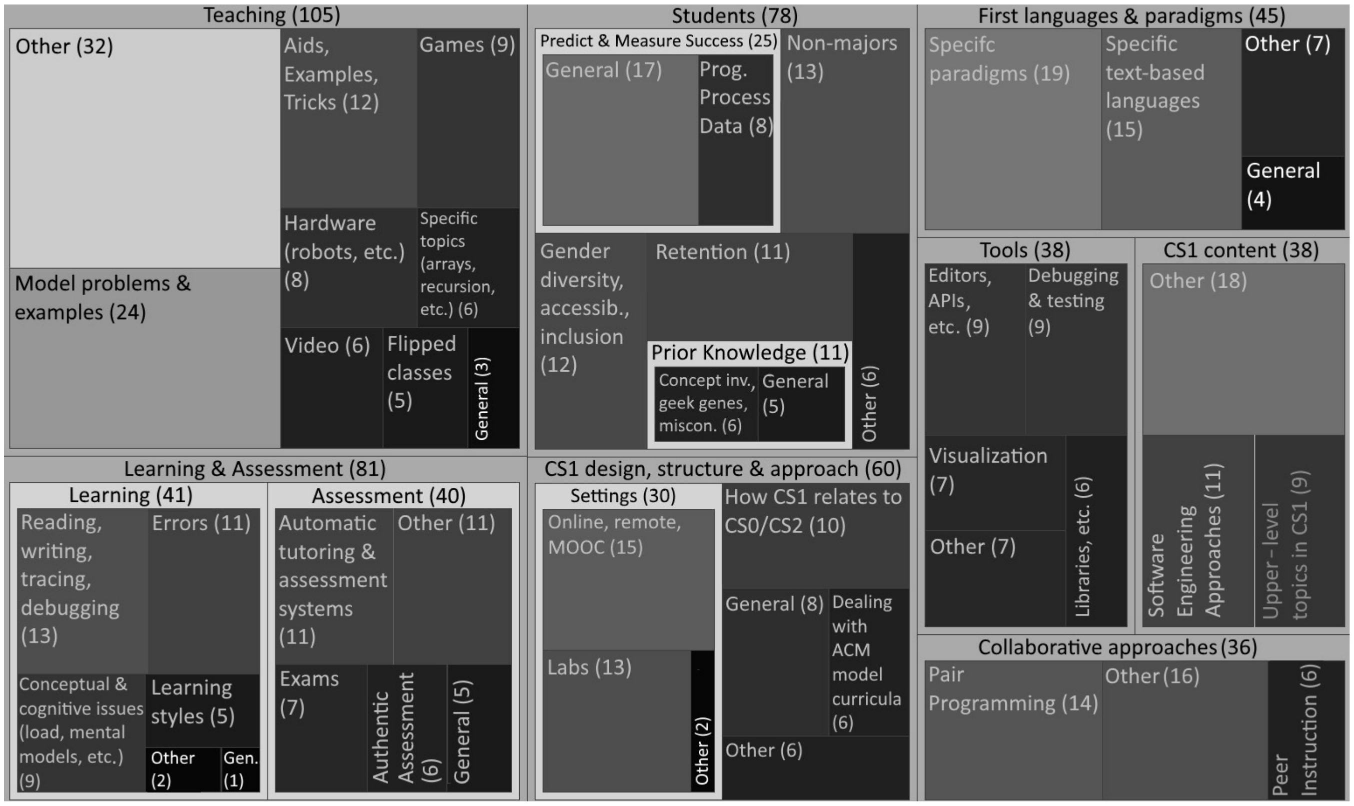
Table 2: Number of hits and percentage of total (ACM DL Guide to Computing Literature) for our search query.

5 THE EVOLUTION OF INTRODUCTORY PROGRAMMING EDUCATION RESEARCH

In this section we provide a view of the introductory programming research presented at the Symposium, followed by discussions on the evolution of each of the eight top-level categories we identified. We also discuss interesting subcategories.

Figure 1 shows a TreeMap of all 481 papers categorized into 8 top-level categories and 54 subcategories. Some top-level categories have a subcategory called 'General', reserved for papers that are broad in scope and represent the whole top-level category as opposed to a specific subcategory. Additionally, if we had a subcategory with less than five papers (other than a 'General' subcategory), we aggregated these into a single 'Other' subcategory. For instance, the top-level category *First languages & paradigms* has a subcategory called *Other* containing papers from any *First languages & paradigms* papers that were in subcategories with less than five papers. Thus, the only subcategories in Figure 1 that may have less than five papers are *General* and *Other* subcategories.

Figure 2 shows the evolution of each of the eight top-level categories over the last five decades. To compensate for the fact that the overall number of papers published in each decade has steadily increased, we normalized the number of papers per decade by dividing the number of papers per top-level category in each decade by the total number of papers in our dataset published in each given decade. This broadly shows the evolution of these eight top-level categories over the last 50 years. In the following subsections we discuss trends and a selection of notable developments/papers in each of these top-level categories, and some subcategories.



Teaching (105)
Other (32)
Aids, Examples, Tricks (12)
Games (9)
Hardware (robots, etc.) (8)
Specific topics (arrays, recursion, etc.) (6)
Model problems & examples (24)
Video (6)
Flipped classes (5)
General (3)
Students (78)
Predict & Measure Success (25)
General (17)
Prog. Process Data (8)
Non-majors (13)
Gender diversity, accessib., inclusion (12)
Retention (11)
Prior Knowledge (11)
Concept inv., geek genes, miscon. (6)
General (5)
Other (6)
First languages & paradigms (45)
Specifc paradigms (19)
Specific text-based languages (15)
Other (7)
General (4)
Tools (38)
Editors, APIs, etc. (9)
Debugging & testing (9)
Visualization (7)
Other (7)
Libraries, etc. (6)
CS1 content (38)
Other (18)
Software Engineering Approaches (11)
Upper-level topics in CS1 (9)
Learning & Assessment (81)
Learning (41)
Reading, writing, tracing, debugging (13)
Errors (11)
Conceptual & cognitive issues (load, mental models, etc.) (9)
Learning styles (5)
Other (2)
Gen. (1)
Assessment (40)
Automatic tutoring & assessment systems (11)
Other (11)
Exams (7)
Authentic Assessment (6)
General (5)
CS1 design, structure & approach (60)
Settings (30)
Online, remote, MOOC (15)
Labs (13)
Other (2)
How CS1 relates to CS0/CS2 (10)
General (8)
Dealing with ACM model curricula (6)
Other (6)
Collaborative approaches (36)
Pair Programming (14)
Other (16)
Peer Instruction (6)

Figure 1: TreeMap of 481 papers in 8 categories and 54 subcategories. The area of each rectangle is proportional to the number of papers in each topic area. More details are discussed in Section 5.

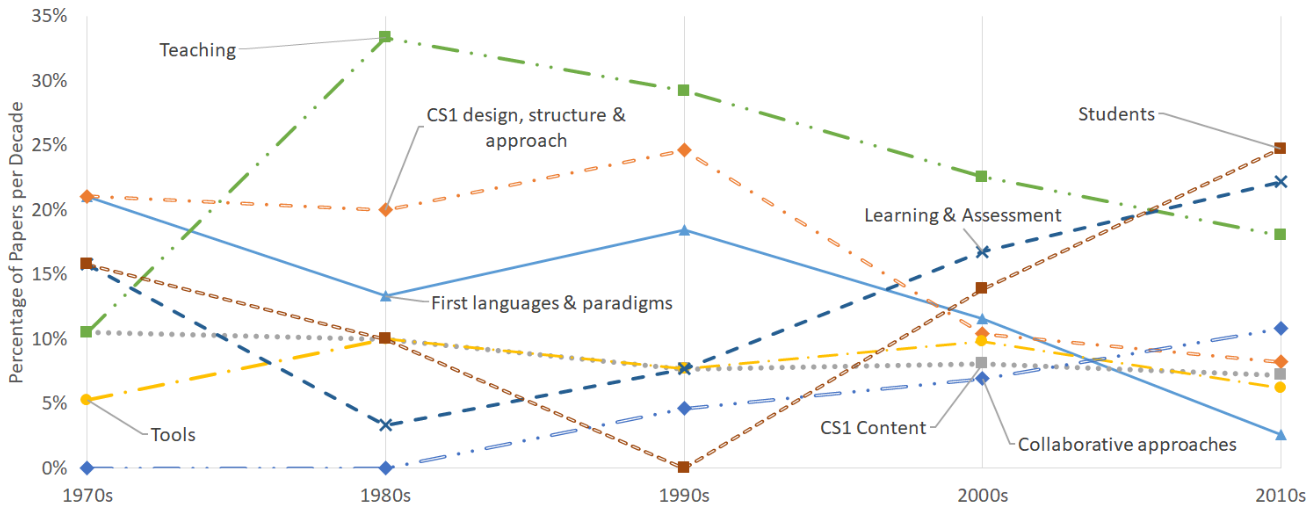


Figure 2: 50-year trends of the 8 top-level categories in introductory programming at the SIGCSE Technical Symposium. To compensate for the fact that the overall number of papers published in each decade has steadily increased, we normalized the number of papers per decade by dividing the number of papers per top-level category in each decade by the total number of papers in our dataset published in each given decade.

5.1 First Languages & Paradigms

	'70s	'80s	'90s	'00s	'10s
General languages & paradigms				2	2
Specific paradigms		1	7	10	1
Specific text-based languages		3	5	6	1
Other	4			2	1

Table 3: Number of papers in the First Languages & Paradigms category.

Table 3 shows the subcategories in this category and the corresponding number of papers published in each decade. As can be seen in Figure 2, interest in this area has slowly declined since the 1990s. Other than language/paradigm choice, one of the most discussed topics in this category has been objects first approaches [12]. At the 2018 Symposium, Simon et al. presented a survey of language choice in CS1 courses in Australasian and UK universities, revealing that the most popular teaching language is Java, followed by Python and C [39]. Interestingly, at the 2004 Symposium, Roberts called for educators to "take responsibility for breaking the cycle of rapid obsolescence [of teaching languages] by developing a stable and effective collection of Java-based teaching resources that will meet the needs of the computer science community" [37, p. 115]. Whether or not this call was met, and whether this was predictive of the prominence of Java reported by Simon et al. is beyond the scope of this paper, but it is interesting to note that the Roberts paper was published in 2004, near a peak in teaching language interest, and that in the current decade, interest has waned considerably, perhaps due to stability brought on by the popular use of Java and Python. Other factors that may have served to draw attention away from the language/paradigm debate recently include the 2004 adoption of Java for the APCS exam [37], which has remained unchanged since, and the evolution of ACM/IEEE model curricula—CC2001 categorized introductory languages by paradigm, while the CS2013 Fundamental Programming Concepts knowledge unit only identifies concepts common to all programming paradigms [23].

5.2 CS1 Design, Structure & Approach

	'70s	'80s	'90s	'00s	'10s
General design; structure; approach	3	1	2		2
How CS1 relates to CS0 or CS2				4	6
Dealing with ACM model curricula		1	3	2	
Physical Settings					
> Online, remote or MOOC delivery			2	8	5
> Labs		1	9	3	
> Other physical settings		1			1
Other	1	2		1	2

Table 4: Number of papers in the CS1 Design, Structure & Approach category.

Table 4 shows the subcategories in this category and the corresponding number of papers published in each decade. Papers in the *Other* category included those on learning outcomes, the impact of hardware decisions and cost-effective methods of teaching CS1. Interest in this category has also waned since the turn of the century, as shown in Figure 2.

It is noteworthy that interest in *Labs* peaked in the 1990s, and *Online, remote or MOOC delivery* peaked a decade or so later. The apparent fall-off in work on remote delivery mechanisms could be due to the advent of new conferences specifically addressing these topics, leading to a migration in where such research is presented. Papers *Dealing with ACM model curricula* peaked in the 1990s, while *How CS1 relates to CS0 and CS2* featured in ten papers, all appearing after 2002 (with four of these since 2016). Perhaps related to this, there has been relatively recent interest in completely rethinking the structure and approach of CS1 itself, as evidenced by a 2010 paper by Hertz titled "What do CS1 and CS2 Mean? Investigating Differences in the Early Courses" [19], and Harvard's 'reinventing' of CS1 resulting in their popular CS50 course, which was redesigned due to two problems that they hypothesized the course suffered from: poor perception and a dated design [31]. It is interesting to note that the idea of redesigning the CS1 course is not new; Cherniak delivered a paper at the 1976 Symposium on reconsidering the CS1 approach [10]. It is noteworthy that despite being presented over 40 years ago, this paper discussed high school programming syllabi. Other notable 'early mentions' were liberal arts colleges in 1980 [15] and delivering CS1 online in 1999 [9].

5.3 CS1 Content

	'70s	'80s	'90s	'00s	'10s
Upper-level topics in CS1				5	4
Software engineering approaches	1	1	4	3	2
Other	1	2	1	6	8

Table 5: Number of papers in the CS1 Content category.

Table 5 shows the subcategories in this category and the corresponding number of papers published in each decade. As seen in Figure 2, interest in this category has remained relatively flat, while the subcategories have varied individually. Topics in the *Other* subcategory include data science approaches, CS+X [5], and media computation [42], which have only appeared in the last two decades. Papers on *Software engineering approaches* [21] have been published in each of the five decades, while papers on *Upper-level topics in CS1* [16] have all appeared after 2000.

5.4 Tools

	'70s	'80s	'90s	'00s	'10s
Editors, APIs, etc.	1		2	2	4
Libraries, etc.			1	5	
Visualization			2	1	4
Debugging & testing		2		6	1
Other		1		3	3

Table 6: Number of papers in the Tools category.

Table 6 shows the subcategories in this category and the corresponding number of papers published in each decade. As seen in Figure 2, similar to *CS1 Content*, interest in this category has remained relatively flat over the decades. As noted in Section 5.1, most of the discussion on CS1 languages has converged on Java and Python, and this is somewhat reflected in the *Tools* category, where several tools for Python [14, 18] and Java [35] have been introduced.

Reis and Cartwright [35] also addressed the ongoing debate of pedagogical vs. 'industry relevant' tools, a theme also reflected in *First languages & paradigms*. Most of the papers on *Libraries* focus on GUI and graphics or games. It is noteworthy that several papers that focused on C or C++ tools were in the *Debugging & testing* and *Visualization* subcategories. It is also noteworthy that not all tools are for development. For instance, Kumar developed *Epplets* to help students solve Parsons Puzzles [26], and the Blackbox project [8] provides tools and a large programming dataset for researchers.

5.5 Collaborative Approaches

	'70s	'80s	'90s	'00s	'10s
Pair programming				5	9
Peer instruction				1	5
Other			3	6	7

Table 7: Number of papers in the Collaborative Approaches category.

Table 7 shows the subcategories in this category and the corresponding number of papers published in each decade. Papers in the *Other* subcategory included topics such as peer code review, studio-based instruction (which includes pedagogical code review) [22], POGIL [20], and group projects. As can be seen in Figure 2, interest in this area has risen dramatically since the turn of the century.

It is arguable that *Pair programming* [33] and *Peer instruction* [40] could be in the same subcategory but after an extensive look at the papers involved and searching the ACM Digital Library for these topics, we decided that they warrant separate subcategories. In fact, none of the six *Peer instruction* papers mentioned *pair programming*, and only one of the 14 *Pair programming* papers mentioned *peer instruction*, and the one that did, discussed the topics as separate approaches [1]. Beck et al. provided a discussion on cooperative learning vs. pair programming [3], and we found one paper categorized in *Other* which, in summarizing the importance of peer learning principles, suggested that 'peer instruction' is a more general term than 'pair programming', 'peer testing' or 'peer review' [17]. Regardless, collaborative approaches to teaching CS1 have steadily increased in popularity since 2000.

5.6 Teaching

	'70s	'80s	'90s	'00s	'10s
General teaching		1		1	1
Model problems & exercises		4	9	10	1
Specific topics (arrays, recursion, etc.)			1	2	3
Games				6	3
Hardware (robots, etc.)				8	
Aids, examples & tricks		2	3	4	3
Flipped approaches					5
Video				2	4
Other	2	3	6	6	15

Table 8: Number of papers in the Teaching category.

Table 8 shows the subcategories in this category and the corresponding number of papers published in each decade. Papers in the *Other* category included those on teaching assistants and mentors, feedback, plagiarism, and many more topics with less than 5 papers each. We chose these subcategories carefully - for instance, *Model problems & exercises* was not chosen as a subcategory of *Learning & Assessment* as by-and-large these are not used as assessment, but as teaching devices. Similarly, we saw *Games*, and *Hardware (robots, etc.)* as teaching devices. Despite *Teaching* being the category with the largest number of papers (105), Figure 2 shows that activity in this area peaked in the 1980s when these papers accounted for 33% of all papers that decade. Since then activity has steadily declined to 18% in this decade. It is worth noting that despite this, the category had the highest percentage of papers per decade in the 1980s, 1990s, and 2000s, and only this decade was surpassed by *Students* and *Learning & Assessment*. Several interesting trends are apparent in Table 8. *Games* first appeared in our data in 2006 [27], but is the third largest subcategory, behind *Model problems & exercises* [7], and *Aids, examples & tricks*. *Hardware (robots, etc.)* first appeared in our data in 2001, *Aids, examples & tricks* has been somewhat constant, and *Flipped approaches* [28] first appeared in 2013.

5.7 Learning & Assessment

	'70s	'80s	'90s	'00s	'10s
General learning					1
Conceptual or cognitive issues			2	4	3
Learning styles				1	4
Reading, writing, tracing & debugging	1	1		5	6
Errors				3	8
Other learning					2
General assessment				4	1
Automatic tutoring & assessment systems	1		1	3	6
Authentic assessment			2	3	1
Exams				2	5
Other assessment	1			4	6

Table 9: Number of papers in the Learning & Assessment category.

Table 9 shows the subcategories in this category and the corresponding number of papers published in each decade. As can be seen in Figure 2 and Table 9, this category has seen steady, sharp growth since the 1980s, moving from 3% to 22% of papers per decade (second only to *Students*). It is important to note however, that as our exclusion criteria eliminates papers less than three pages in length, we exclude tracks such as panels. Therefore our data does not accurately represent initiatives like "Nifty Assignments" which started as a panel at the Symposium in 1999 and is now a track of its own (nifty.stanford.edu). A number of subcategories have shown remarkable interest since the turn of the century, including *Reading, writing, tracing & debugging* [13], how students deal with *Errors* [4, 32], and *Learning styles*, including achievement goals and mastery learning [47], and *Conceptual or cognitive issues* which revealed papers on topics such as mental models [30].

5.8 Students

Table 10 shows the subcategories in this category and the corresponding number of papers published in each decade. Figure 2

	'70s	'80s	'90s	'00s	'10s
Non-majors	2			3	8
Retention		1		5	5
Gender, diversity, inclusion & accessibility				5	7
Prior knowledge				2	3
> Concept inventories, geek genes, misconceptions				1	5
Predicting & measuring success		2		5	10
> Programming process data				2	6
Other	1			1	4

Table 10: Number of papers in the Students category.

shows that this category went from 16% in the 1970s to 0% in the 1990s to 25% today (the most popular category). Several insights can be drawn from Table 10. It is interesting to note that *Non-majors* were considered as early as 1973 [38], yet interest has only recently gathered pace. *Gender, diversity, inclusion & accessibility* has only appeared since 2004, when Rich et al. proposed a CS1 course explicitly aimed at non-majors and women [36]. In 2004, Valentine noted that only half of Symposia between 1984 and 2004 had any 'appearances' on women in computer science. *Prior knowledge* including *Concept inventories* [43], *geek genes* (or the non-existence thereof) and *Misconceptions* [24] have also only appeared since 2000. In 2017, Kirkpatrick et al. introduced an alternative CS1 for students with prior programming knowledge [25]. *Predicting & measuring success* [6, 46] and research on *Programming process data* [2] have also only appeared since the turn of the century. It is interesting to note that concerns about large enrollments appeared as early as 1974 [11], while *Retention*, dating back as far as 1984, has shown significant interest more recently [34].

6 THREATS TO VALIDITY

We note several threats to validity, but do not believe that these significantly detract from achieving our goals set out in Section 1. Firstly, our search query may not be completely representative of the literature, but it is similar to a recent survey with a much wider scope [29], and as discussed in Section 4, we believe it is broadly representative of the wider literature. Also, if carried out by other researchers, the categories that emerged may have been different.

We acknowledged the "significant difficulty of reliably coding academic papers" [41, p. 223] and as this is not a systematic literature review, we allowed categories to emerge as we began classification. However as outlined in Section 3.1 our categorization proved to be acceptably reproducible. Next, in the interest of being succinct, we employed a many-to-one mapping of papers to categories (each paper has exactly one category, but a category may have more than one paper). Thus, we did not allow cross-categorization, which may have provided for a richer (but probably more complex) analysis and discussion. Additionally, our trends are coarsely based on decades. A finer resolution may have provided more subtle trends.

In excluding papers less than three pages in length, we excluded tracks such as panels, and therefore initiatives such as Nifty Assignments which provide valuable contributions. Finally, we remind the reader that we have specifically researched CS1 papers presented at the SIGCSE Technical Symposium only, and that that the trends observed may not necessarily be reflective of the broader trends of the computing education community.

7 CONCLUSION

We categorized 481 CS1 papers presented during the first 50 years of the SIGCSE Technical Symposium. We have endeavored to adopt best practice to categorize these papers in the interest of being fair and objective, and believe that the picture we present of the evolution of introductory programming research during the first 50 years of the Symposium is useful in terms of meeting the research goals set out in Section 1. Future work on a larger scale could address some of the threats identified in Section 6. Our analysis revealed several important trends and highlights the fact that CS1 research is not only focused on programming. We also situated the Symposium in the wider literature, revealing that it occupies a significant space in the CS1 literature.

This work provides a checkpoint for introductory programming research going forward into the second 50 years of the SIGCSE Technical Symposium and will hopefully provide insights for future research on how we teach novices and how they learn to program.

REFERENCES

[1] Onni Aarne, Petrus Peltola, Juho Leinonen, and Arto Hellas. 2018. A Study of Pair Programming Enjoyment and Attendance Using Study Motivation and Strategy Metrics. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 759–764. https://doi.org/10.1145/3159450.3159493

[2] Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 522–527. https://doi.org/10.1145/2676723.2677258

[3] Leland L. Beck, Alexander W. Chizhik, and Amy C. McElroy. 2005. Cooperative Learning Techniques in CS1: Design and Experimental Evaluation. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, USA, 470–474. https://doi.org/10.1145/1047344.1047495

[4] Brett A. Becker, Kyle Goslin, and Graham Glanville. 2018. The Effects of Enhanced Compiler Error Messages on a Syntax Error Debugging Test. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 640–645. https://doi.org/10.1145/3159450.3159461

[5] Tanya Berger-Wolf, Boris Igic, Cynthia Taylor, Robert Sloan, and Rachel Poretsky. 2018. A Biology-themed Introductory CS Course at a Large, Diverse Public University. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 233–238. https://doi.org/10.1145/3159450.3159538

[6] Susan Bergin and Ronan Reilly. 2005. Programming: Factors That Influence Success. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, USA, 411–415. https://doi.org/10.1145/1047344.1047480

[7] Kevin Bierre, Phil Ventura, Andrew Phelps, and Christopher Egert. 2006. Motivating OOP by Blowing Things Up: An Exercise in Cooperation and Competition in an Introductory Java Programming Course. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '06)*. ACM, New York, NY, USA, 354–358. https://doi.org/10.1145/1121341.1121452

[8] Neil Christopher Charles Brown, Michael Kölling, Davin McCall, and Ian Utting. 2014. Blackbox: A Large Scale Repository of Novice Programmers' Activity. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 223–228. https://doi.org/10.1145/2538862.2538924

[9] Jacobo Carrasquel. 1999. Teaching CS1 On-line: The Good, the Bad, and the Ugly. In *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education (SIGCSE '99)*. ACM, New York, NY, USA, 212–216. https://doi.org/10.1145/299649.299758

[10] Bob Cherniak. 1976. Introductory Programming Reconsidered - a User-oriented Approach. In *Proceedings of the ACM SIGCSE-SIGCUE Technical Symposium on Computer Science and Education (SIGCSE '76)*. ACM, New York, NY, USA, 65–68. https://doi.org/10.1145/800107.803449

[11] Richard W. Conway. 1974. Introductory Instruction in Programming. In *Proceedings of the Fourth SIGCSE Technical Symposium on Computer Science Education (SIGCSE '74)*. ACM, New York, NY, USA, 6–10. https://doi.org/10.1145/800183.810430

[12] Stephen Cooper, Wanda Dann, and Randy Pausch. 2003. Teaching Objects-first in Introductory Computer Science. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)*. ACM, New York, NY, USA, 191–195. https://doi.org/10.1145/611892.611966

[13] Stephen H. Edwards. 2004. Using Software Testing to Move Students from Trial-and-error to Reflection-in-action. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, USA, 26–30. https://doi.org/10.1145/971300.971312

[14] Stephen H. Edwards, Daniel S. Tilden, and Anthony Allevato. 2014. Pythy: Improving the Introductory Python Programming Experience. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 641–646. https://doi.org/10.1145/2538862.2538977

[15] Robert J. Ellison. 1980. A Programming Sequence for the Liberal Arts College. In *Proceedings of the Eleventh SIGCSE Technical Symposium on Computer Science Education (SIGCSE '80)*. ACM, New York, NY, USA, 161–164. https://doi.org/10.1145/800140.804628

[16] Patrick Garrity, Timothy Yates, Richard Brown, and Elizabeth Shoop. 2011. WebMapReduce: An Accessible and Adaptable Tool for Teaching Map-reduce Computing. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 183–188. https://doi.org/10.1145/1953163.1953221

[17] Alessio Gaspar, Joni Torsella, Nora Honken, Sohum Sohoni, and Colin Arnold. 2016. Differences in the Learning Principles Dominating Student-Student vs. Student-Instructor Interactions While Working on Programming Tasks. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 255–260. https://doi.org/10.1145/2839509.2844627

[18] 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 (SIGCSE '13)*. ACM, New York, NY, USA, 579–584. https://doi.org/10.1145/2445196.2445368

[19] Matthew Hertz. 2010. What Do "CS1" and "CS2" Mean?: Investigating Differences in the Early Courses. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 199–203. https://doi.org/10.1145/1734263.1734335

[20] Helen H. Hu and Tricia D. Shepherd. 2014. Teaching CS 1 with POGIL Activities and Roles. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 127–132. https://doi.org/10.1145/2538862.2538954

[21] Christopher Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. 2009. Integrating Pedagogical Code Reviews into a CS 1 Course: An Empirical Study. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*. ACM, New York, NY, USA, 291–295. https://doi.org/10.1145/1508865.1508972

[22] Christopher Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. 2010. Does Studio-based Instruction Work in CS 1?: An Empirical Comparison with a Traditional Approach. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 500–504. https://doi.org/10.1145/1734263.1734432

[23] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Technical Report. New York, NY, USA. 999133.

[24] Lisa C. Kaczmarczyk, Elizabeth R. Petrick, J. Philip East, and Geoffrey L. Herman. 2010. Identifying Student Misconceptions of Programming. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 107–111. https://doi.org/10.1145/1734263.1734299

[25] Michael S. Kirkpatrick and Chris Mayfield. 2017. Evaluating an Alternative CS1 for Students with Prior Programming Experience. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 333–338. https://doi.org/10.1145/3017680.3017759

[26] Amruth N. Kumar. 2018. Epplets: A Tool for Solving Parsons Puzzles. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 527–532. https://doi.org/10.1145/3159450.3159576

[27] Scott Leutenegger and Jeffrey Edgington. 2007. A Games First Approach to Teaching Introductory Programming. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*. ACM, New York, NY, USA, 115–118. https://doi.org/10.1145/1227310.1227352

[28] Kate Lockwood and Rachel Esselstein. 2013. The Inverted Classroom and the CS Curriculum. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 113–118. https://doi.org/10.1145/2445196.2445236

[29] Andrew Luxton-Reilly, Simon, Ibrihim 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 of the 2018 ITiCSE Conference on Working Group Reports (ITiCSE-WGR '18)*. ACM, New York, NY, USA. https://doi.org/10.1145/3293881.3295779 *in press*.

[30] Linxiao Ma, John Ferguson, Marc Roper, and Murray Wood. 2007. Investigating the Viability of Mental Models Held by Novice Programmers. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*. ACM, New York, NY, USA, 499–503. https://doi.org/10.1145/1227310.1227481

[31] David J. Malan. 2010. Reinventing CS50. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 152–156. https://doi.org/10.1145/1734263.1734316

[32] Guillaume Marceau, Kathi Fisler, and Shriram Krishnamurthi. 2011. Measuring the Effectiveness of Error Messages Designed for Novice Programmers. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 499–504. https://doi.org/10.1145/1953163.1953308

[33] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The Effects of Pair-programming on Performance in an Introductory Programming Course. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '02)*. ACM, New York, NY, USA, 38–42. https://doi.org/10.1145/563340.563353

[34] Leo Porter and Daniel Zingaro. 2014. Importance of Early Performance in CS1: Two Conflicting Assessment Stories. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 295–300. https://doi.org/10.1145/2538862.2538912

[35] Charles Reis and Robert Cartwright. 2004. Taming a Professional IDE for the Classroom. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, USA, 156–160. https://doi.org/10.1145/971300.971357

[36] Lauren Rich, Heather Perry, and Mark Guzdial. 2004. A CS1 Course Designed to Address Interests of Women. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, USA, 190–194. https://doi.org/10.1145/971300.971370

[37] Eric Roberts. 2004. The Dream of a Common Language: The Search for Simplicity and Stability in Computer Science Education. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, USA, 115–119. https://doi.org/10.1145/971300.971343

[38] Gerard Salton. 1973. Introductory Programming at Cornell. In *Proceedings of the Third SIGCSE Technical Symposium on Computer Science Education (SIGCSE '73)*. ACM, New York, NY, USA, 18–20. https://doi.org/10.1145/800010.808068

[39] Simon, Raina Mason, Tom Crick, James H. Davenport, and Ellen Murphy. 2018. Language Choice in Introductory Programming Courses at Australasian and UK Universities. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 852–857. https://doi.org/10.1145/3159450.3159547

[40] Beth Simon, Michael Kohanfars, Jeff Lee, Karen Tamayo, and Quintin Cutts. 2010. Experience Report: Peer Instruction in Introductory Computing. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 341–345. https://doi.org/10.1145/1734263.1734381

[41] Andreas Stefik, Stefan Hanenberg, Mark McKenney, Anneliese Andrews, Srinivas Kalyan Yellanki, and Susanna Siebert. 2014. What is the Foundation of Evidence of Human Factors Decisions in Language Design? An Empirical Study on Programming Language Workshops. In *Proceedings of the 22Nd International Conference on Program Comprehension (ICPC 2014)*. ACM, New York, NY, USA, 223–231. https://doi.org/10.1145/2597008.2597154

[42] Allison Elliott Tew, Charles Fowler, and Mark Guzdial. 2005. Tracking an Innovation in Introductory CS Education from a Research University to a Two-year College. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)*. ACM, New York, NY, USA, 416–420. https://doi.org/10.1145/1047344.1047481

[43] Allison Elliott Tew and Mark Guzdial. 2010. Developing a Validated Assessment of Fundamental CS1 Concepts. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 97–101. https://doi.org/10.1145/1734263.1734297

[44] David W. Valentine. 2004. CS Educational Research: A Meta-analysis of SIGCSE Technical Symposium Proceedings. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '04)*. ACM, New York, NY, USA, 255–259. https://doi.org/10.1145/971300.971391

[45] Troy Vasiga. 2002. What Comes After CS 1 + 2: A Deep Breadth Before Specializing. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (SIGCSE '02)*. ACM, New York, NY, USA, 28–32. https://doi.org/10.1145/563340.563350

[46] Christopher Watson, Frederick W.B. Li, and Jamie L. Godwin. 2014. No Tests Required: Comparing Traditional and Dynamic Predictors of Programming Success. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 469–474. https://doi.org/10.1145/2538862.2538930

[47] Daniel Zingaro, Michelle Craig, Leo Porter, Brett A. Becker, Yingjun Cao, Phill Conrad, Diana Cukierman, Arto Hellas, Dastyni Loksa, and Neena Thota. 2018. Achievement Goals in CS1: Replication and Extension. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 687–692. https://doi.org/10.1145/3159450.3159452