Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Malmi, Lauri; Sheard, Judy; Kinnunen, Päivi; Simon, Simon; Sinclair, Jane

Computing Education Theories: What Are They and How Are They Used?

Published in:
International Computing Education Research Conference (ICER '19), August 12–14, 2019, Toronto, ON, Canada

DOI:
10.1145/3291279.3339409

Published: 11/08/2019

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Malmi, L., Sheard, J., Kinnunen, P., Simon, S., & Sinclair, J. (2019). Computing Education Theories: What Are They and How Are They Used? In International Computing Education Research Conference (ICER '19), August 12–14, 2019, Toronto, ON, Canada (pp. 187-197). ACM. https://doi.org/10.1145/3291279.3339409

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

Computing Education Theories: What Are They and How Are They Used?

Lauri Malmi
Aalto University, Finland
lauri.malmi@aalto.fi

Judy Sheard
Monash University, Australia
judy.sheard@monash.edu

Päivi Kinnunen
Aalto University, Finland
paivi.kinnunen@aalto.fi

Simon
University of Newcastle, Australia
simon@newcastle.edu.au

Jane Sinclair
University of Warwick, UK
j.e.sinclair@warwick.ac.uk

ABSTRACT

In order to mature as a research field, computing education research (CER) seeks to build a better theoretical understanding of how students learn computing concepts and processes. Progress in this area depends on the development of computing-specific theories of learning to complement the general theoretical understanding of learning processes. In this paper we analyze the CER literature in three central publication venues – ICER, ACM Transactions of Computing Education, and Computer Science Education – over the period 2005–2015. Our findings identify new theoretical constructs of learning computing that have been published, and the research approaches that have been used in formulating these constructs. We identify 65 novel theoretical constructs in areas such as learning/understanding, learning behaviour/strategies, study choice/orientation, and performance/progression/retention. The most common research methods used to devise new constructs include grounded theory, phenomenography, and various statistical models. We further analyze how a number of these constructs, which arose in computing education, have been used in subsequent research, and present several examples to illustrate how theoretical constructs can guide and enrich further research. We discuss the implications for the whole field.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

computing education, theory, theoretical construct, research, method

ACM Reference Format:
Lauri Malmi, Judy Sheard, Päivi Kinnunen, Simon, and Jane Sinclair. 2019. Computing Education Theories: What Are They and How Are They Used?. In *International Computing Education Research Conference (ICER '19), August 12–14, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3291279.3339409

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICER '19, August 12–14, 2019, Toronto, ON, Canada
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6185-9/19/08…$15.00
https://doi.org/10.1145/3291279.3339409

1 INTRODUCTION

Computing education research (CER) seeks to build deep understanding of the complex phenomena and processes involved in teaching and learning computing. The field is maturing [80] in terms of Fensham's criteria [21] for identifying a research discipline. In the early days most published works were small case studies presenting novel teaching innovations and practices, often called *practice papers* [22]. However, during the past 10–15 years increasing numbers of papers can be characterized as research papers in the sense that they have clear research questions and rigorous settings for empirical data collection and analysis, and are informed by some theoretical framework [49, 80]. These theoretical frameworks are typically adopted from the social sciences, particularly educational science and psychology, in addition to some originating in the computing sciences themselves [55].

One of Fensham's criteria for an independent field of science is that the field builds its own domain-specific theories. Malmi et al. [54] explored the CER literature, covering 308 papers from the ICER conference and two prestigious journals, ACM Transactions of Computing Education (TOCE) and Computer Science Education (CSEd) from 2005–2011. Their main interest was the extent to which papers published in these venues were building on some theory, model or framework (TMF), and in which disciplines these constructs had been developed. As part of their work they identified 23 TMFs originating in CER itself, only two of which were originally published in their data pool.

Since that work was carried out, the field has undergone substantial growth, as seen, for example, in the number of papers submitted to ICER and the corresponding decline in the acceptance rate[1]. Further attention has also been paid to the significance of theories in CER. Lishinski et al. [49] explored papers published in CSEd and ICER in 2012–2015. They mainly considered methodological rigor, but also looked at the use of theories from outside CER. They reported:

> "A significantly higher proportion of articles in our sample made use of outside theory than in the results presented by Malmi et al, suggesting that the field is increasingly reaching into other disciplines to frame and interpret studies with respect to previous research in learning theory" (p167).

On the other hand, Nelson and Ko [59] present a critical view of the role of theories in CER, acknowledging that theories can inform

[1] See ACM Digital library for details.

research design, but noting that in some cases theories may inhibit the design of educational innovations. They call for more work to build CER domain-specific theories. It is clear that we cannot expect researchers in social sciences to be able to develop theories on, for example, how students learn programming or data structures and algorithms. Deep understanding of the subject domain subject is needed for such work, and we should seek to establish fruitful collaboration between the fields.

Inspired by these works, we seek to extend our understanding of the role of theories in CER. Responding to the call of Lishinski et al., we focus our investigation specifically on domain-specific theories. Our work also responds to the question raised by Malmi et al. [54] concerning whether CER is developing its own theories. We cover more recent literature than they did, but also focus on theories that were first published explicitly in our data pool. Our specific research questions are:

(1) What theoretical constructs concerning learning and education in computing have been published in ICER, CSEd, and JERIC/TOCE[2] in 2005–2015?
(2) In what areas are these constructs situated?
(3) What methods were used to derive these constructs?
(4) In what different ways have these constructs been used or extended in further research?

The last question is particularly interesting, because theoretical work has limited value unless it is used to inform further research, which is possible in several different ways.

2 ROLES OF THEORY

There is a wealth of theories used in CER, and a wealth of ways in which theories can be used in the field. As a young research discipline, computing education borrows theories and methodologies from the social sciences, which are themselves not homogeneous when it comes to ways of conceptualizing and using theories. Mjøset [68] distinguishes three theoretical traditions in social science: standard/natural science, social philosophy, and the contextualized view. Each tradition comes with its own background, customs, and methodological guidelines. For instance, the natural science tradition sees theory as law-oriented construction, whereas the social-philosophical tradition sees it as reconstructive or deconstructive. Across these different theoretical traditions, it is difficult to offer a single, clear definition of theory and of its role in computing education. In this paper we have decided to adopt the rather pragmatic and broad notion of theory proposed by Pears et al. [63, p2]: a theory is "a way to reason about the nature and structure of observable learning phenomena, and their operation and interrelationships".

Theory can contribute to educational research in many different ways. Suppes [84] describes a number of roles for theories in educational research: theories help us to notice what is important and essential in a phenomenon; theories guide us to recognize the complexity of a phenomenon that at first glance may seem to be simple; theories may provide tools for making predictions; and, most importantly, theories help us to overcome the triviality of bare empiricism that at its worst provides neither a theoretical understanding of a phenomenon nor useful practical guidelines. Similarly, in mathematics education Niss [60] suggests that theories can serve as an overarching framework to guide the viewpoint from which we look at a phenomenon; that they may offer a way of organising a set of specific observations; and that they may offer ideas for a research methodology or a suitable terminology to discuss a phenomenon.

To remain useful in these roles, theories often need to evolve and mature. Tellings [68, 86] suggests four ways that different theories can be combined. In *reduction*, one theory is redefined in terms of another theory; in *synthesis*, integration of theories leads to totally new insights; in *horizontal addition*, theories that cover different aspects of one domain are combined to give a more holistic understanding of a phenomenon; and in *vertical addition*, different theories that describe different stages of development are stacked on top of one another. This evolution of theories is of interest in this project as one of our research questions addresses how emerging computing education theories have been used or extended in subsequent studies.

3 METHOD

Our research investigates new theoretical constructs that were initially published in ICER, CSEd, or TOCE between 2005 and 2015. These venues were selected because they are known for publishing high-quality research in the form of long papers that facilitate the incorporation and discussion of theoretical frameworks. We considered only full papers published at these venues, not other works such as editorials and short papers. Because we are interested in the possible impact of domain-specific theoretical constructs on subsequent research, we bounded our search at 2015, estimating that three years beyond the original publication of a construct was sufficient time to identify the potential impact. This gave us a data set of 540 papers: 192 from ICER, 172 from CSEd, and 176 from TOCE.

The terminology concerning theories is used loosely in the CER literature. Terms such as 'theory', 'model', and 'framework' are frequently used with no clear distinction between them, and very few theories have established recognizable names such as Jadud's EQ [34] or Robins's learning edge momentum [70]. Moreover, interesting contributions can be reported without using any of these terms; for example, studies applying phenomenography generally call their empirical results 'outcome spaces'. Recognizing that a strict definition of theory would be likely to reduce our search results too much, we defined the concept *theoretical construct* as a theory, model, framework, or instrument developed through application of some rigorous empirical or theoretical approach. To implement our search we developed the following guidelines for identifying theoretical constructs.

First, in quantitative research, we included statistical models, such as regression, path analysis, factor analysis, structural equation modelling, and clustering, which have been generated from the data to explain the relationships between identified concepts. Typically these include mathematical formulas and/or graphical presentations of the model. We did not include results that present statistical evidence for accepting individual hypotheses, unless the results

[2]The Journal of Educational Resources in Computing (JERIC) was replaced in 2009 by ACM Transactions on Computing Education (TOCE), bringing about a clear change in the journal's profile. Our data pool includes JERIC papers from 2005–2008 and TOCE papers from 2009–2015, but for convenience we refer to them all as TOCE papers.

were further developed to build a more comprehensive explanation of the investigated phenomenon.

Second, in qualitative research we included grounded theories, phenomenographical outcome spaces, and other data-driven categorizations that seek to generate a higher-level abstract description of the data. We excluded simple lists of categories – which often result from qualitative content analysis – if there was no clear discussion of their relationships.

Third, we included explanatory models of other types that can be identified as taxonomies, typologies, figures, formulas, or sets of categories and the relationships between them. These may be generated by analyzing data, derived from other theoretical frameworks adapted to address a specific problem domain, or developed through logical argumentation.

Fourth, we included validated instruments for the measurement of particular theory-based concepts.

Finally, we were interested only in theoretical constructs that were clearly developed to address aspects of computing education, even if they might later be applied in other fields – as have, for example, zone of proximal flow [3] and learning edge momentum [70].

Our study comprised four stages to address our four research questions.

3.1 Identifying theoretical constructs (TCs)

Our definition of *theoretical constructs* (TCs) comprises a potentially vast number of forms and labels. Rather than conducting a systematic search of papers using a predetermined set of terms, we decided that we would need to closely read each paper in our data set to identify any new constructs.

Two members of the research team independently read each paper and recorded the occurrence of any new theoretical construct reported in the work. They then discussed any differences and reached agreement as to whether a paper recorded the development of a new theoretical construct. At the conclusion of this process 65 new theoretical constructs had been identified from the 540 papers. We call the papers where they were published the *source papers*.

3.2 Determining areas of focus of TCs

The theoretical constructs that we found have been developed to address highly varied research questions in many different research contexts. Seeking a more holistic picture of the results, we categorized the TCs into broad thematic areas, using inductive content analysis to identify the main focus area of each construct. This could reveal whether the constructs were related to learning or understanding some topic, for example, or to students' progress, or to students' attitudes. The categories were devised by a member of the research team through further reading of the TCs' source papers. Two other team members then re-read the papers and reviewed the categories, leading to minor adjustments.

We also wanted to examine the pool of TCs from the point of view of instructional processes. For this we used the didactic triangle [35] as a lens to investigate how widely the TCs address the central actors and their relationships in the process. The didactic triangle is a simple model, drawn as a triangle, whose vertexes are the three most relevant actors, aspects of teaching and learning (teacher, student, and learning content), and whose edges are the relationships between the actors (teacher-student, teacher-content, and student-content). Studying and learning take place within the student-content relationship and are influenced by teachers' pedagogical actions. Kinnunen et al. [40] added a further relationship, students' perceptions of a pedagogical action such as student feedback. Together these actors and relationships cover the central aspects of the instructional process. Our goal with this analysis was to find out whether the TCs we had identified focused on particular aspects of the instructional process or covered it more broadly.

The analysis based on the triangle was carried out by two researchers who independently categorized all source papers, then compared their results and reached agreement. For this research the categorization was based solely on the paper's use of the theoretical construct. As a TC can address more than one aspect within the triangle, such as the teacher's pedagogical actions and the student-content relationship, two or more aspects could be recorded as the result for a single TC.

3.3 Identifying methods used to develop TCs

We sought to determine the foundations of the theoretical constructs and how they had been developed. One member of the research team assessed whether each construct was completely new or was developed from an existing construct, and identified the approach and methods used to develop the construct. This analysis was then reviewed by another team member and agreement was reached.

3.4 Exploring how TCs are used

In the final stage of our work we investigated how some of the theoretical constructs we identified have been used, seeking to find out whether theoretical constructs are actually used to guide further research or are simply cited as related work. Using Google Scholar, we searched for each original source paper in which a TC was presented, and counted the papers listed under 'cited by', which we call *citation papers*. Overall we found 3200 citation papers, clearly rendering complete analysis beyond our resources. We therefore limited this phase of the analysis to two selected focus areas, learning/understanding and learning behavior/strategies, and for most source papers in these focus areas we downloaded every citation paper that we could access. For those few source papers that have a large number of citation papers, we considered only the citations from recent years. These reductions left us with 516 citation papers to examine.

Within the citation papers we searched for mentions of the source paper, or of the TC if it had a name. We then tagged each citation paper according to whether 1) the source paper was simply described as related work, possibly with an explanation of the TC; 2) the citation paper applied the TC, for example, for developing a data collection or analysis method or for explaining results; 3) the citation paper modified or extended the original TC; or 4) the citation paper reported on a study to validate the TC or applied it in a novel context.

We found several cases where the citation paper did not cite the source paper at all, presumably due to errors in Google Scholar. There were also a few cases in which we were unable to access the

citation paper, and a few citation papers written in languages other than English, which we were unable to analyze.

4 RESULTS

4.1 What TCs have been developed?

Our analysis of publications in ICER, CSEd, and TOCE from 2005 to 2015 found 65 papers reporting new TCs. Table 1 shows the distribution of these source papers across the venues, along with a summary of their citation numbers. We examined the data by year of publication to check for possible trends, but none were apparent.

Table 1: Papers reporting the development of a theoretical construct (TC) in ICER, CSEd, and TOCE from 2005 to 2015 and their citations

Venue	Total papers	Papers reporting a new TC	Average citations[a]	Citation range[a]
ICER	192	38 (20%)	53	1–288
CSEd	172	21 (12%)	42	5–182
TOCE	176	6 (3%)	51	5–150

a. According to Google Scholar, 31 March 2019

4.2 What are the areas of focus of TCs?

As indicated in section 3.2, we grouped the 65 TCs into related areas of focus. Table 2 shows these focus areas, along with the number of TCs in each, and the source paper for each TC. Almost a third of the TCs focus on learning/understanding, followed by study choice/orientation, performance/progression/retention, and learning behavior/strategies. The remaining categories each have three or four TCs.

The table shows, not surprisingly, that theory development has been most active in areas dealing with aspects of students' learning process, learning results, and other relationships between students and the learning content or study field. But students do not learn in isolation: teachers have a major impact on the learning process. To investigate how well the TCs cover the teaching aspect, we conducted a closer analysis based on the didactic triangle [35], which provides a clear theoretical framework for identifying various aspects of the instructional process.

Recalling that a single TC can cover more than one aspect of the triangle, we identified 75 instances of a TC covering some aspect of the triangle. Two-thirds of TCs cover one aspect only, while most of the rest cover two or more aspects. The distribution of these aspects is very skewed. In 52 cases of the 75, the TC focuses on the student-content relationship. Half of these concern students' understandings or attitudes to the content, and the other half their learning activities and learning results or motivational changes. The next most common aspect (10 cases) addresses teachers' pedagogical actions, followed by six cases focusing on the content itself, three cases for the teacher-content relationship, two cases for the teacher-student relationship, and one case each for teachers and student feedback. None of the 65 TCs addresses students as such. Finally, three TCs [40, 55, 79] were not related to the didactic triangle as their source papers analyzed computing education research literature in general.

4.3 How were TCs developed?

We identified 18 different approaches to developing the TCs in our data set, as shown in Table 3.

We identified eight theoretical constructs that were developed by adapting an existing theory and five that were developed by extending an existing theory. The remaining 80% were new TCs that may have drawn on but were not built on or adapted from existing theories.

In more than a third of the cases the TC was developed by following the formal qualitative methodologies of grounded theory or phenomenography. The most common quantitative approach, used in 12% of the cases, was regression. In more than a third of the cases a combination of methods was used, typically involving combinations of literature review, argumentation, empirical studies, qualitative analysis, and quantitative analysis. Of course argumentation is an integral part of every analysis, but we recorded it only if it played a substantial role in the analysis in which the construct was derived.

Table 4 lists the 20 TCs in the area of learning/understanding and the approach that was used to derive each of them. Space limitations mean that we cannot to do the same for the other areas of focus, but we hope that these 20 cases provide a reasonable idea of the nature of the findings.

4.4 How are TCs used?

Of the 516 citation papers that we analyzed, the great majority (91%) make no use at all of the TCs from the source papers. In many cases (65%) the citation typically refers to the source paper in a group of references, or at most writes a sentence or two describing what the source paper is about. Readers of the citation paper gain little from such a reference, and in most cases we could not determine whether the TC was genuinely relevant to the citing paper. Far less frequently (21%), we found citation papers that summarize the main points of the source paper's contribution and explain its relevance to the citation paper. And in very rare cases (5%) we found critical discussion on the TC or a comparison of it with other works. Examples of this include critiques on Robins's learning edge momentum theory [70] by Luxton-Reilly [52] and Petersen et al. [64]. The latter, for example, accepts Robins's contention that programming concepts are highly interdependent, but cites evidence to question the assumption that in typical programming courses these concepts are introduced in a clear linear sequence.

Notwithstanding the overall picture, we did find a number of cases (10%) where theoretical constructs were used to inform further research. In a couple of cases these papers also included critical discussion or comparison of the TC with other works. We shall present several examples of how this was done in order to illustrate how future research in CER can benefit from theories developed in CER. We do not distinguish here between cases where the continued development is carried out by the original authors or, more rarely, by somebody else.

4.4.1 Using theory to discuss results. Niss [60] notes that theories can be used to *explain* observed phenomena. However, a theory or model is always a hypothesis or simplification of reality, and the same observations may have different interpretations in real life. In the natural sciences empirical observations are widely used to

Table 2: Areas of focus of the theoretical constructs and the source papers

Area of Focus	Count	Source Papers
learning/understanding	20	[3, 6, 8, 9, 11, 19, 31, 46, 50, 58, 70, 73, 74, 77, 82, 83, 87, 89, 96, 97]
study choice/orientation	8	[7, 18, 26, 28, 29, 47, 51, 72]
performance/progression/retention	7	[4, 5, 25, 39, 48, 65, 85]
learning behavior/strategies	6	[10, 14, 23, 33, 43, 99]
contents/curriculum/learning goals	4	[17, 24, 56, 57]
emotion/beliefs/attitudes/self-efficacy	4	[41, 42, 71, 76]
teaching/pedagogical content knowledge	4	[12, 32, 81, 92]
assessment/self-assessment	3	[1, 53, 78]
computing education research	3	[40, 55, 79]
errors/misconceptions	3	[34, 62, 93]
perceptions of computer science/computing	3	[27, 75, 98]
Total	**65**	

Table 3: Approaches used to develop the theoretical constructs

Approach	Count	Source Papers
phenomenography	12	[6, 8–10, 12, 19, 39, 82, 83, 87, 89, 92]
grounded theory or grounded theory based	10	[23, 27–29, 40, 42, 47, 56, 75, 97]
regression	8	[4, 5, 7, 18, 26, 71, 72, 85]
argumentation + empirical study	7	[14, 33, 50, 70, 77–79]
adapted an existing theory + empirical study	6	[1, 3, 11, 57, 58, 76]
empirical study	5	[17, 34, 43, 46, 98]
extended an existing theory + argumentation	3	[40, 55, 81]
extended an existing theory + empirical study	2	[31, 74]
literature analysis + argumentation	2	[25, 73]
literature analysis + argumentation + empirical study	2	[48, 93]
adapted an existing theory + grounded theory	1	[62]
adapted an existing theory + structural equation modeling	1	[51]
apply an existing theory + empirical study	1	[99]
delphi study	1	[24]
literature analysis + empirical study	1	[65]
path analysis	1	[96]
used a theory + argumentation	1	[53]
used a theory + literature analysis + argumentation	1	[32]
Total	**65**	

support or counter a theory; but in education the settings may be more complex, with numerous intervening contextual variables, and strong conclusions can rarely be justified. However, this does not undermine the search for better explanations for empirical data. Different TCs can be used as lenses to interpret the data in order to deepen and broaden our understanding of the case itself and of potentially conflicting factors in existing explanations.

This point is nicely illustrated by Porter and Zingaro [66], who analyzed peer instruction and exam data to explore relationships between in-class assessments and performance in mid-term and final exams in CS1. They analyzed correlations between the factors during a course, with the goal of identifying good predictors for final results. Considering student performance in weeks 3 and 4 of the course, they note that their findings could provide equal support for the competing theories of the 'geek gene', learning edge momentum, and stumbling points. But moving further on in the course, they observe that their findings for later weeks appear troublesome for each of those three theories. Such discussion encourages researchers to build new settings, where different theories can be tested against the same data, so as to tease out the factors behind the observed conflicts between theories and data.

4.4.2 Using theory to predict results. Another use of theory, closely related to the previous one, is predicting what can happen [60], which is a natural way to test theories and their explanatory power.

Carter et al. [14] analyzed students' programming process and developed a normalized programming state model (NPSM) to describe various states of program correctness (syntactic or semantic) and transitions between these states. They speculated that the relative time spent in correct and incorrect states could be an indicator for students' overall course performance. This makes intuitive sense: students who are struggling with syntax errors are likely to have their programs in incorrect states for most of the time, whereas students who are working to resolve runtime errors are likely to

Table 4: Theoretical constructs with a focus on learning and understanding

Year	Theoretical construct	Approach	Citations[a]
2005	Refining/extending Good's schema to extract information about novice programmers' comprehension of concurrent programs and confidence in the captured knowledge [31]	extended an existing theory + empirical	8
2005	A model of factors that are important for non-majors' learning of programming [96]	path analysis	148
2005	Categorizing novice programming students' understanding of what it means to learn to program [19]	phenomenography	107
2006	Categorizing students' learning aims in an international project-based course [6]	phenomenography	23
2006	Role plan analysis model to evaluate students' mental models of programming concepts [11]	adapted an existing theory + empirical	15
2006	Categorizing novice programming students' understanding of what it means to learn how to program and their understanding of program correctness [83]	phenomenography	27
2008	Hierarchical model of programming skills [50]	argumentation + empirical	182
2009	Categorizing students' understanding of a Java interface and their understanding of software systems [8]	phenomenography	6
2010	The notion of the learning edge momentum effect to explain the patterns of introductory programming course outcomes [70]	argumentation + empirical	161
2011	Categorizing practitioners' understanding of an object-oriented program [89]	phenomenography	1
2012	Categorizing graduating students' understanding of class diagrams [9]	phenomenography	10
2012	The notion of threshold skills [74]	extended an existing theory + empirical	23
2013	Categorizing novice programmers' perceptions of learning through visual program simulation [82]	phenomenography	15
2013	Extending the definition of threshold concepts [73]	literature + argumentation	10
2013	Zones of proximal flow pedagogical framework to promote intrinsic motivation and leverage student learning experiences [3]	adapted an existing theory + empirical	36
2013	Progression of early computational thinking (PECT) model for understanding and assessing computational thinking of primary school students [77]	argumentation + empirical	92
2014	Computer science cognitive load component survey instrument for measuring cognitive load in an introductory programming context [58]	empirical	50
2014	Categorizing graduating students' understanding of what it means to 'produce a design' [87]	phenomenography	7
2014	Taxonomy for describing how students understand recursion [46]	empirical	14
2014	Model of students' computational thinking processes while learning to program [97]	grounded theory	5

a. According to Google Scholar, 31 March 2019

be working with and testing programs that are correct, at least syntactically. The study compares the explanatory power of NPSM with the predictions generated by two previous theories, Jadud's EQ [34] and Watson score [95]. NPSM explained a larger share of variance than the other measures. However, their results for the other measures differed significantly from earlier published results, and Carter et al. discuss possible reasons for this. Such discussion is important to help the research community to better understand the limitations and contextual factors for different TCs. In this case the principal contextual factors are the programming language and environment: NPSM was initially developed for C++ programming using Visual Studio. Further work by Richards and Hunt [69] investigates the model in a different context to determine whether it can be applied with BlueJ despite some clear differences between the two environments. Their work is still inconclusive with regard to the predictive power of NPSM, but it is important to investigate whether the initial reported success was tied to a specific context.

In later work, Carter et al. investigated more closely the transition patterns between different states of NPSM [13], finding differences between the observed patterns of low- and high-performing students. In further work [15] they combined the previous model with a measure of students' social participation in a social programming environment and compared this to students using a traditional environment. This investigation was motivated by social learning theory [44], which emphasizes regular participation in a learning community. The results suggest that for all models (EQ, Watson score, and NPSM), complementing them with a measure of social activity improves their predictive power on course performance. The improved model, SNPSM, is thus a nice example of Telling's horizontal addition of theories: covering more aspects of the phenomenon improves explanatory power [86].

4.4.3 Using theory to inform pedagogy and testing the results. Theories are valuable if they can be used to inform practical pedagogical decisions to improve learning results or learning experience.

Hoda and Andreae [30] applied learning edge momentum (LEM)[3] to inform pedagogy. Essentially, LEM claims that CS1 programming concepts are tightly connected, and that failure or success to understand a concept may have a cumulative negative or positive effect on the learning of closely related concepts. This results in the supposedly bimodal distributions often seen in CS1 final exams.

In their LEM-based pedagogical approach for CS1, Hoda and Andreae apply the following strategies: 1) minimizing early complexities in the course; 2) minimizing dependencies between early components of the course; 3) maximizing chances of mastery of the early concepts and skills; and 4) maximizing opportunities for early recovery. For example, they revised assignments to reduce their interdependency, and introduced intermediate assignments to enable mastery of individual constructs and to build bridges between concepts. Their results were promising, with considerable reduction of failure rates in all student cohorts from several different backgrounds.

While we do not claim that their work provides compelling evidence to support LEM, the paper serves as an interesting example of how theory can inform practice and how practice can be used to test and possibly validate theory.

Another example of using TCs to inform pedagogy is the work of Thota [90], which discusses how phenomenographical TCs can be used in course design. She explains how phenomenography can inform us about what students find relevant in the learning situation, and how variation theory can be used to design experiences that support students' awareness of the differences in understandings among their peers when dealing with the same object of learning. She provides a concrete example of course design to demonstrate how the guidelines can be implemented.

4.4.4 Using a TC as a data analysis framework. In section 4.4.1 we discussed how theory can be used to interpret results. A closely related case is the use of a TC as a tool to support data analysis.

Table 4 includes several TCs that categorize students' understandings of some topic or concept; these can be used as tools for further research.

Thomas et al. [87] analyzed final-year students' understanding of software design by giving them a requirement specification for a 'super alarm clock' and asking them to produce a design that others could work from. They classified the resulting designs into a phenomenographical outcome space with six categories ranging from nothing to a complete design. Later they explored the design skills of a cohort at the midpoint of their studies in order to investigate how students' design skills develop over time [88]. They replicated their previous study with a different design task but using the same categorization to analyze results. Thus their earlier work served as a qualitative measure of designs, allowing comparison of results between studies. This is a good example of using a phenomenographical outcome space in subsequent research.

Another example involves the theory of zones of proximal flow (ZPF) [3, 36], which illustrates Telling's *synthesis* of theories [86] by combining Vygotsky's zone of proximal development [94] and Csikszentmihalyi's flow concept [16]. The general idea of this theory is that students make optimal progress when they work in an area that lies between anxiety (task is too hard) and boredom (task is too easy). They should be given tasks that challenge them while not being too hard. This approach has had an important role in scalable game design pedagogy. As explained by Repenning et al. [67, p9], "The project ... is likely to push students to their threshold of understanding (the Zone of Proximal Development), but ... they manage to learn the relevant concepts in an optimal way that is highly engaging."

In further work, Basawapatna et al. [2] explore how ZPF can be leveraged in real time to support formative feedback; that is, how to follow student activities in the classroom and provide feedback based on indications that a student is lost and needs assistance. This was implemented with positive results.

Illustrating TCs in the form of validated instruments, Morrison et al. [58] took an instrument initially developed to measure cognitive load [45], adapted it for computer science, and validated it in the new context. In terms of Telling's terminology [86] discussed in section 2, this is an example of *reduction*. The new instrument has subsequently been used by other researchers for measuring cognitive load in different contexts, such as worked examples in programming [100], Parsons problems [20], and collaborative algorithms labs [91].

5 DISCUSSION

Overall we found a rich selection of TCs developed in computing education and first presented in ICER, CSEd, or TOCE. They focus mainly on the student-content relationship of the didactic triangle, which is no surprise: previous research based on the didactic triangle has shown that this relationship is the most commonly researched area, not only in computing education [40] but also in engineering education [38] and science education [37]. We did find some theoretical work addressing other aspects, but the results suggest that there is scope for more work to be done. Could we build theories or models, for example, to explain how computing teachers design and implement their pedagogical actions or how they teach particular computing topics or skills? Concerning students, instead of collecting and reporting course feedback, could we seek to build a deeper understanding of how students experience various pedagogical actions? These are just examples to illustrate how the didactic triangle could be used to generate research questions or TCs in areas other than the student-content relationship.

5.1 Using TCs

Our findings indicate that a considerable amount of theoretical work is indeed carried out in CER, which from Fensham's perspective [21] is one indication of a field's independence. CER papers are, of course, also published in many other venues, and our survey reveals only a part of the big picture. However, we believe that the selected venues are likely to cover a substantial share of the overall work.

It is clear that theories are of limited value if they are not used. Here our findings are less promising. Our detailed investigation of this aspect covers only two of the 11 different focus areas, though those two comprise a substantial part of the TC pool. Here we found, as reported in Section 4.4, that almost all citation papers merely mention the reference to a source paper, and among these it is not always clear why the reference was given at all. Perhaps the authors

[3] While LEM has been critiqued, as discussed in section 4.4, it has not been broadly rejected.

considered the TC related in some way to their own work while not being close enough to merit more discussion; or perhaps it was some other contribution of the source paper that warranted its inclusion in the reference list. There was seldom sufficient information in the citation papers to permit such judgements to be made.

This raises the question of why there is so little use of TCs to inform other work. Our current data does not support analysis of this phenomenon, but we can speculate on some of the reasons.

First, educational settings for research are notoriously complex and any theoretical constructs are likely to include hidden assumptions about course content, teaching methods or tools, student population, teachers' competencies, even personality, etc. These factors are seldom well enough described in the paper to allow readers to judge whether the context of the contribution is sufficiently similar to their own settings. This might be remedied somewhat if the TC authors were to make their contextual assumptions as visible as possible, and explicitly discuss the transferability of their construct to other settings.

Second, some TCs might be difficult to implement outside the original research context. For example, qualitative categorizations of data, such as phenomenographical outcome spaces, can provide deep insight into the investigated phenomenon among the target cohort. However, applying them in other settings, such as to quantify the distribution of different conceptions in a different cohort, may entail serious challenges. The work required to analyze and categorize a larger data pool may be overwhelming, and there may be a dearth of suitable methodological tools, such as validated instruments, for carrying out the categorization. Analysis may face problems with inter-rater reliability, as reported by Parham-Mocello and Ernst [61] when they tried to apply the design quality categories from Thomas et al. [87]. These challenges are less serious for the original authors of the source papers due to their experience in the analysis work, so they are better able to apply their outcome spaces in their further research.

The difficulties reported by Parham-Mocello and Ernst [61] encouraged us to look more broadly at the impact of qualitative work on subsequent research in computing education. We selected for further analysis the nine papers in either learning/understanding or learning behavior/strategies (Table 2) that were developed using a phenomenographical approach (Table 3), and examined all 200+ citation papers of those nine source papers. We did not find a single case in which someone other than the original researchers had used the TC as a data collection and analysis framework, and we found only few cases in which the TC was used in discussion of results or in building new pedagogical solutions. This is somewhat concerning, because there is presumably some valuable work being done, yet its potential for re-use is not being realized.

In contrast, validated instruments for data collection and analysis are clearly easier to apply and are being applied by other researchers, as discussed above in relation to the cognitive load survey instrument of Morrison et al. [58]. It remains an open question whether it is feasible and practical to develop validated instruments, based on qualitative TCs, that would support easy quantitative exploration of students' understandings or experiences. Such progress would be a valuable contribution to the field.

Finally, we suggest that the low rate of adoption of TCs might be due to uncertainty about the overall role of theories in computing education, or about appropriate ways to apply theories. We hope that some of the examples we have given will assist the CER community in this regard. More examples would be even more helpful, but cannot be reported here due to space limitations.

In summary, the current impact of TCs on continuing research seem modest. We expected to find more evidence of the impact of TCs on educational practice, but again we were disappointed, finding only a small number of papers that directly apply TCs to inform educational practice. However, this observation probably reflects the visibility problem of TCs rather than their actual impact. We hope that there are many teachers who apply results from literature in their teaching, but do not write papers about their experiences and observations. It is not possible to evaluate the scope of this invisible impact. Even when papers do report on innovations built upon some TC, they might not report the link explicitly enough for us to have identified it.

5.2 Limitations

Our work covers only three of the publication venues in which TCs might be published, and the limited time span from 2005 to 2015: this prevents us from building a complete picture of the development of theoretical constructs in computing education research. The exclusion of papers published since 2015 was a deliberate decision to support our goal of analyzing the usage of theories in subsequent research. We acknowledge the influence of personal interpretations in deciding which constructs to include and which to exclude, and in how the TCs were categorized. We increased the trustworthiness of the results by using three experienced qualitative researchers in this process (two in the selection phase and three in the categorization phase) to read the papers and negotiate a consensus for each paper and TC. Our definition for a TC was deliberately loose to allow us to discover and make visible a greater amount of work that could benefit the CER community.

6 CONCLUSION

In this paper our goal has been to survey a substantial portion of the computing education research literature over 11 years, identifying the development of theories that focus on building deeper insights into the complex world of teaching and learning computing concepts and processes. We believe that there is much of value in this work, which builds domain-specific knowledge to support the future work of our growing research community, and we want to enhance its visibility. We have identified a total of 65 theoretical constructs, noting how each was derived. We have grouped them into 11 different focus areas, most of which deal with describing or modelling the rich relationship between students and computing concepts. We provide a number of different examples to show how these constructs have been used to inform further research or educational practice, and we hope that such examples will inspire other researchers to consider using or extending existing theoretical work – when they are not developing theories of their own.

In future work, we plan to expand our pool of source papers by considering more computing education venues, and to extend our detailed analysis of discipline-specific theoretical constructs beyond the learning/understanding and learning behavior/strategies areas of focus that we have addressed in this paper.

REFERENCES

[1] Satu Alaoutinen. 2012. Evaluating the effect of learning style and student background on self-assessment accuracy. *Computer Science Education* 22, 2 (2012), 175–198.

[2] Ashok Ram Basawapatna, Alexander Repenning, and Kyu Han Koh. 2015. Closing the cyberlearning loop: enabling teachers to formatively assess student programming projects. In *46th ACM Technical Symposium on Computer Science Education (SIGCSE 2015)*. ACM, 12–17.

[3] Ashok R Basawapatna, Alexander Repenning, Kyu Han Koh, and Hilarie Nickerson. 2013. The zones of proximal flow: guiding students through a space of computational thinking skills and challenges. In *Ninth International Computing Education Research Conference (ICER 2013)*. ACM, 67–74.

[4] Jens Bennedsen and Michael E Caspersen. 2005. An investigation of potential success factors for an introductory model-driven programming course. In *First International Computing Education Research Workshop (ICER 2005)*. ACM, 155–163. https://doi.org/10.1145/1089786.1089801

[5] Susan Bergin and Ronan Reilly. 2006. Predicting introductory programming performance: a multi-institutional multivariate study. *Computer Science Education* 16, 4 (2006), 303–323. https://doi.org/10.1080/08993400600997096

[6] Anders Berglund and Anna Eckerdal. 2006. What do CS students try to learn? Insights from a distributed, project-based course in computer systems. *Computer Science Education* 16, 3 (2006), 185–195. https://doi.org/10.1080/08993400600912368

[7] Sylvia Beyer. 2014. Why are women underrepresented in computer science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education* 24, 2-3 (2014), 153–192. https://doi.org/10.1080/08993408.2014.963363

[8] Jonas Boustedt. 2009. Students' understanding of the concept of interface in a situated context. *Computer Science Education* 19, 1 (2009), 15–36. https://doi.org/10.1080/08993400902819980

[9] Jonas Boustedt. 2012. Students' different understandings of class diagrams. *Computer Science Education* 22, 1 (2012), 29–62. https://doi.org/10.1080/08993408.2012.665210

[10] Ilona Box. 2009. Toward an understanding of the variation in approaches to analysis and design. *Computer Science Education* 19, 2 (2009), 93–109. https://doi.org/10.1080/08993400902909674

[11] Pauli Byckling and Jorma Sajaniemi. 2006. A role-based analysis model for the evaluation of novices' programming knowledge development. In *Second International Computing Education Research Workshop (ICER 2006)*. ACM, 85–96. https://doi.org/10.1145/1151588.1151602

[12] Angela Carbone, Linda Mannila, and Sue Fitzgerald. 2007. Computer science and IT teachers' conceptions of successful and unsuccessful teaching: a phenomenographic study. *Computer Science Education* 17, 4 (2007), 275–299. https://doi.org/10.1080/08993400701706586

[13] Adam Scott Carter and Christopher David Hundhausen. 2017. Using programming process data to detect differences in students' patterns of programming. In *48th ACM Technical Symposium on Computer Science Education (SICSE 2017)*. ACM, 105–110.

[14] Adam S Carter, Christopher D Hundhausen, and Olusola Adesope. 2015. The normalized programming state model: predicting student performance in computing courses based on programming behavior. In *11th International Computing Education Research Conference (ICER 2015)*. ACM, 141–150. https://doi.org/10.1145/2787622.2787710

[15] Adam S Carter, Christopher D Hundhausen, and Olusola Adesope. 2017. Blending measures of programming and social behavior into predictive models of student achievement in early computing courses. *ACM Transactions on Computing Education (TOCE)* 17, 3 (2017), 12.

[16] Mihaly Csikszentmihalyi. 1975. *Beyond boredom and anxiety: the experience of play in work and leisure*. San Francisco, CA: Jossey-Bass.

[17] Quintin Cutts, Sarah Esper, Marlena Fecho, Stephen R Foster, and Beth Simon. 2012. The abstraction transition taxonomy: developing desired learning outcomes through the lens of situated cognition. In *[Eighth] International Computing Education Research Conference (ICER 2012)*. ACM, 63–70. https://doi.org/10.1145/2361276.2361290

[18] Toni Downes and Dianne Looker. 2011. Factors that influence students' plans to take computing and information technology subjects in senior secondary school. *Computer Science Education* 21, 2 (2011), 175–199. https://doi.org/10.1080/08993408.2011.579811

[19] Anna Eckerdal, Michael Thuné, and Anders Berglund. 2005. What does it take to learn 'programming thinking'?. In *First International Computing Education Research Workshop (ICER 2005)*. ACM, 135–142. https://doi.org/10.1145/1089786.1089799

[20] Barbara Jane Ericson. 2018. *Evaluating the effectiveness and efficiency of Parsons problems and dynamically adaptive Parsons problems as a type of low cognitive load practice problem*. Ph.D. Dissertation. Georgia Institute of Technology.

[21] Peter F Fensham. 2004. *Defining an identity – the evolution of science education as a field of research*. Springer.

[22] Sally Fincher and Marian Petre. 2004. *Computer science education research*. CRC Press.

[23] Sue Fitzgerald, Beth Simon, and Lynda Thomas. 2005. Strategies that students use to trace code: an analysis based in grounded theory. In *First International Computing Education Research Workshop (ICER 2005)*. ACM, 69–80. https://doi.org/10.1145/1089786.1089793

[24] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey L Herman, Lisa Kaczmarczyk, Michael C Loui, and Craig Zilles. 2010. Setting the scope of concept inventories for introductory computing subjects. *ACM Transactions on Computing Education (TOCE)* 10, 2 (2010), 5.

[25] Mark Guzdial. 2013. Exploring hypotheses about media computation. In *Ninth International Computing Education Research Conference (ICER 2013)*. ACM, 19–26. https://doi.org/10.1145/2493394.2493397

[26] Mark Guzdial, Barbara J Ericson, Tom McKlin, and Shelly Engelman. 2012. A statewide survey on computing education pathways and influences: factors in broadening participation in computing. In *[Eighth] International Computing Education Research Conference (ICER 2012)*. ACM, 143–150. https://doi.org/10.1145/2361276.2361304

[27] Michael Hewner. 2013. Undergraduate conceptions of the field of computer science. In *Ninth International Computing Education Research Conference (ICER 2013)*. ACM, 107–114. https://doi.org/10.1145/2493394.2493414

[28] Michael Hewner. 2014. How CS undergraduates make course choices. In *Tenth International Computing Education Research Conference (ICER 2014)*. ACM, 115–122. https://doi.org/10.1145/2632320.2632345

[29] Michael Hewner and Mark Guzdial. 2011. How CS majors select a specialization. In *Seventh International Computing Education Research Workshop (ICER 2011)*. ACM, 11–18. https://doi.org/10.1145/2016911.2016916

[30] Rashina Hoda and Peter Andreae. 2014. It's not them, it's us! Why computer science fails to impress many first years. In *Sixteenth Australasian Computing Education Conference (ACE 2014)*. Australian Computer Society, Inc, 159–162.

[31] Connor Hughes, Jim Buckley, Chris Exton, and Des O'Carroll. 2005. Towards a framework for characterising concurrent comprehension. *Computer Science Education* 15, 1 (2005), 7–24. https://doi.org/10.1080/08993400500056522

[32] Petri Ihantola, Ville Karavirta, Ari Korhonen, and Jussi Nikander. 2005. Taxonomy of effortless creation of algorithm visualizations. In *First International Computing Education Research Workshop (ICER 2005)*. ACM, 123–133. https://doi.org/10.1145/1089786.1089798

[33] Essi Isohanni and Maria Knobelsdorf. 2010. Behind the curtain: students' use of VIP after class. In *Sixth International Computing Education Research Workshop (ICER 2010)*. ACM, 87–96. https://doi.org/10.1145/1839594.1839610

[34] Matthew C Jadud. 2006. Methods and tools for exploring novice compilation behaviour. In *Second International Computing Education Research Workshop (ICER 2006)*. ACM, 73–84.

[35] Pertti Kansanen. 2003. Studying – the realistic bridge between instruction and learning. An attempt to a conceptual whole of the teaching-studying-learning process. *Educational Studies* 29, 2-3 (2003), 221–232.

[36] Kristian Kiili, Sara De Freitas, Sylvester Arnab, and Timo Lainema. 2012. The design principles for flow experience in educational games. *Procedia Computer Science* 15 (2012), 78–91.

[37] Päivi Kinnunen, Jarkko Lampiselkä, Veijo Meisalo, and Lauri Malmi. 2016. Research on teaching and learning in physics and chemistry in NorDiNa papers. *NorDina: Nordisk tidsskrift i naturfagdidaktikk* (2016).

[38] P Kinnunen and L Malmi. 2013. Pedagogical focus of recent engineering education research papers. In *SEFI conference*. 16–20.

[39] Päivi Kinnunen, Robert McCartney, Laurie Murphy, and Lynda Thomas. 2007. Through the eyes of instructors: a phenomenographic investigation of student success. In *Third International Computing Education Research Workshop (ICER 2007)*. ACM, 61–72. https://doi.org/10.1145/1288580.1288589

[40] Päivi Kinnunen, Viejo Meisalo, and Lauri Malmi. 2010. Have we missed something? Identifying missing types of research in computing education. In *Sixth International Computing Education Research Workshop (ICER 2010)*. ACM, 13–22. https://doi.org/10.1145/1839594.1839598

[41] Päivi Kinnunen and Beth Simon. 2010. Experiencing programming assignments in CS1: the emotional toll. In *Sixth International Computing Education Research Workshop (ICER 2010)*. ACM, 77–86.

[42] Päivi Kinnunen and Beth Simon. 2012. My program is ok – am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education* 22, 1 (2012), 1–28.

[43] Aditi Kothiyal, Rwitajit Majumdar, Sahana Murthy, and Sridhar Iyer. 2013. Effect of think-pair-share in a large CS1 class: 83% sustained engagement. In *Ninth International Computing Education Research Conference (ICER 2013)*. ACM, 137–144.

[44] Jean Lave and Etienne Wenger. 1991. *Situated learning: legitimate peripheral participation*. Cambridge University Press.

[45] Jimmie Leppink, Fred Paas, Tamara Van Gog, Cees PM van Der Vleuten, and Jeroen JG Van Merrienboer. 2014. Effects of pairs of problems and examples on task performance and different types of cognitive load. *Learning and Instruction* 30 (2014), 32–42.

[46] Colleen M Lewis. 2014. Exploring variation in students' correct traces of linear recursion. In *Tenth International Computing Education Research Conference (ICER 2014)*. ACM, 67–74. https://doi.org/10.1145/2632320.2632355

[47] Colleen M Lewis, Ken Yasuhara, and Ruth E Anderson. 2011. Deciding to major in computer science: a grounded theory of students' self-assessment of ability. In *Seventh International Computing Education Research Workshop (ICER 2011)*. ACM, 3–10. https://doi.org/10.1145/2016911.2016915

[48] Tracy L Lewis, Wanda J Smith, France Bélanger, and K Vernard Harrington. 2008. Are technical and soft skills required? The use of structural equation modeling to examine factors leading to retention in the CS major. In *Fourth International Computing Education Research Workshop (ICER 2008)*. ACM, 91–100. https://doi.org/10.1145/1404520.1404530

[49] Alex Lishinski, Jon Good, Phil Sands, and Aman Yadav. 2016. Methodological rigor and theoretical foundations of CS education research. In *12th International Computing Education Research Conference (ICER 2016)*. ACM, 161–169. https://doi.org/10.1145/2960310.2960328

[50] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships between reading, tracing and writing skills in introductory programming. In *Fourth International Computing Education Research Workshop (ICER 2008)*. ACM, 101–112. https://doi.org/10.1145/1404520.1404531

[51] Andy Luse, Julie A Rursch, and Doug Jacobson. 2014. Utilizing structural equation modeling and social cognitive career theory to identify factors in choice of IT as a major. *ACM Transactions on Computing Education (TOCE)* 14, 3, Article 19 (Sept. 2014), 19 pages. https://doi.org/10.1145/2623198

[52] Andrew Luxton-Reilly. 2016. Learning to program is easy. In *21st Conference on Innovation and Technology in Computer Science Education (ITiCSE 2016)*. ACM, 284–289.

[53] Andrew Luxton-Reilly and Paul Denny. 2010. Constructive evaluation: a pedagogy of student-contributed assessment. *Computer Science Education* 20, 2 (2010), 145–167.

[54] Lauri Malmi, Judy Sheard, Simon, Roman Bednarik, Juha Helminen, Päivi Kinnunen, Ari Korhonen, Niko Myller, Juha Sorva, and Ahmad Taherkhani. 2014. Theoretical underpinnings of computing education research: what is the evidence?. In *Tenth International Computing Education Research Conference (ICER 2014)*. ACM, 27–34. https://doi.org/10.1145/2632320.2632358

[55] Lauri Malmi, Judy Sheard, Simon, Roman Bednarik, Juha Helminen, Ari Korhonen, Niko Myller, Juha Sorva, and Ahmad Taherkhani. 2010. Characterizing research in computing education: a preliminary analysis of the literature. In *Sixth International Computing Education Research Workshop (ICER 2010)*. ACM, 3–12. https://doi.org/10.1145/1839594.1839597

[56] Monica M. McGill. 2012. The curriculum planning process for undergraduate game degree programs in the United Kingdom and United States. *ACM Transactions on Computing Education (TOCE)* 12, 2, Article 7 (April 2012), 47 pages.

[57] Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. 2013. Learning computer science concepts with Scratch. *Computer Science Education* 23, 3 (2013), 239–264. https://doi.org/10.1080/08993408.2013.832022

[58] Briana B Morrison, Brian Dorn, and Mark Guzdial. 2014. Measuring cognitive load in introductory CS: adaptation of an instrument. In *Tenth International Computing Education Research Conference (ICER 2014)*. ACM, 131–138. https://doi.org/10.1145/2632320.2632348

[59] Greg L Nelson and Andrew J Ko. 2018. On use of theory in computing education research. In *2018 ACM Conference on International Computing Education Research (ICER 2018)*. ACM, 31–39. https://doi.org/10.1145/3230977.3230992

[60] Mogens Niss. 2007. The concept and role of theory in mathematics education. *Relating practice and research in mathematics education. Norma* 5 (2007), 97–110.

[61] Jennifer Parham-Mocello and Shannon Ernst. 2017. Analysis of freshmen designs and the correlation to grades. *Journal of Computing Sciences in Colleges* 33, 1 (2017), 186–193.

[62] Thomas H Park, Ankur Saxena, Swathi Jagannath, Susan Wiedenbeck, and Andrea Forte. 2013. Towards a taxonomy of errors in HTML and CSS. In *Ninth International Computing Education Research Conference (ICER 2013)*. ACM, 75–82.

[63] Arnold Pears, Neena Thota, Päivi Kinnunen, and Anders Berglund. 2012. Harnessing theory in the service of engineering education research. In *2012 Frontiers in Education Conference Proceedings*. IEEE, 1–5.

[64] Andrew Petersen, Michelle Craig, and Daniel Zingaro. 2011. Reviewing CS1 exam question content. In *42nd ACM Technical Symposium on Computer Science Education (SIGCSE 2011)*. ACM, 631–636.

[65] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. 2011. Peer instruction: do students really learn from peer discussion in computing?. In *Seventh International Computing Education Research Workshop (ICER 2011)*. ACM, 45–52. https://doi.org/10.1145/2016911.2016923

[66] Leo Porter and Daniel Zingaro. 2014. Importance of early performance in CS1: two conflicting assessment stories. In *45th ACM Technical Symposium on Computer Science Education (SIGCSE 2014)*. ACM, 295–300.

[67] Alexander Repenning, David C Webb, Kyu Han Koh, Hilarie Nickerson, Susan B Miller, Catharine Brand, Ian Her Many Horses, Ashok Basawapatna, Fred Gluck, Ryan Grover, Kris Gutierrez, and Nadia Repenning. 2015. Scalable game design: a strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education (TOCE)* 15, 2 (2015), 11.

[68] Research Council of Norway. 2011. *The role of theory in educational research*. Technical Report. Research Council of Norway.

[69] Brad Richards and Ayse Hunt. 2018. Investigating the applicability of the normalized programming state model to BlueJ programmers. In *18th Koli Calling International Conference on Computing Education Research (Koli Calling 2018)*. ACM, 10.

[70] Anthony Robins. 2010. Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education* 20, 1 (2010), 37–71.

[71] Ma Mercedes T Rodrigo and Ryan SJd Baker. 2009. Coarse-grained detection of student frustration in an introductory programming course. In *Fifth International Computing Education Research Workshop (ICER 2009)*. ACM, 75–80.

[72] Mary Beth Rosson, John M Carroll, and Hansa Sinha. 2011. Orientation of undergraduates toward careers in the computer and information sciences: gender, self-efficacy and social support. *ACM Transactions on Computing Education (TOCE)* 11, 3, Article 14 (Oct. 2011), 23 pages. https://doi.org/10.1145/2037276.2037278

[73] Janet Rountree, Anthony Robins, and Nathan Rountree. 2013. Elaborating on threshold concepts. *Computer Science Education* 23, 3 (2013), 265–289. https://doi.org/10.1080/08993408.2013.834748

[74] Kate Sanders, Jonas Boustedt, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Lynda Thomas, and Carol Zander. 2012. Threshold concepts and threshold skills in computing. In *[Eighth] International Computing Education Research Conference (ICER 2012)*. ACM, 23–30. https://doi.org/10.1145/2361276.2361283

[75] Carsten Schulte and Maria Knobelsdorf. 2007. Attitudes towards computer science – computing experiences as a starting point and barrier to computer science. In *Third International Computing Education Research Workshop (ICER 2007)*. ACM, 27–38. https://doi.org/10.1145/1288580.1288585

[76] Michael James Scott and Gheorghita Ghinea. 2014. Measuring enrichment: the assembly and validation of an instrument to assess student self-beliefs in CS1. In *Tenth International Computing Education Research Conference (ICER 2014)*. ACM, 123–130.

[77] Linda Seiter and Brendan Foreman. 2013. Modeling the learning progressions of computational thinking of primary grade students. In *Ninth International Computing Education Research Conference (ICER 2013)*. ACM, 59–66. https://doi.org/10.1145/2493394.2493403

[78] Judy Sheard, Simon, Angela Carbone, Donald Chinn, Mikko-Jussi Laakso, Tony Clear, Michael de Raadt, Daryl D'Souza, James Harland, Raymond Lister, Anne Philpott, and Geoff Warburton. 2011. Exploring programming assessment instruments: a classification scheme for examination questions. In *Seventh International Computing Education Research Workshop (ICER 2011)*. ACM, 33–38.

[79] Simon. 2007. A classification of recent Australasian computing education publications. *Computer Science Education* 17, 3 (2007), 155–169. https://doi.org/10.1080/08993400701538021

[80] Simon. 2015. *Emergence of computing education as a research discipline*. Ph.D. Dissertation. Aalto University.

[81] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 4, Article 15 (Nov. 2013), 64 pages. https://doi.org/10.1145/2490822

[82] Juha Sorva, Jan Lönnberg, and Lauri Malmi. 2013. Students' ways of experiencing visual program simulation. *Computer Science Education* 23, 3 (2013), 207–238. https://doi.org/10.1080/08993408.2013.807962

[83] Ioanna Stamouli and Meriel Huggard. 2006. Object oriented programming and program correctness: the students' perspective. In *Second International Computing Education Research Workshop (ICER 2006)*. 109–118.

[84] Patrick Suppes. 1974. The place of theory in educational research. *Educational Researcher* 3, 6 (1974), 3–10.

[85] Emily S Tabanao, Ma Mercedes T Rodrigo, and Matthew C Jadud. 2011. Predicting at-risk novice Java programmers through the analysis of online protocols. In *Seventh International Computing Education Research Workshop (ICER 2011)*. ACM, 85–92. https://doi.org/10.1145/2016911.2016930

[86] Agnes Tellings. 2001. Eclecticism and integration in educational theories: a metatheoretical analysis. *Educational Theory* 51, 3 (2001), 277–292.

[87] Lynda Thomas, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Kate Sanders, and Carol Zander. 2014. Graduating students' designs: through a phenomenographic lens. In *Tenth International Computing Education Research Conference (ICER 2014)*. ACM, 91–98. https://doi.org/10.1145/2632320.2632353

[88] Lynda Thomas, Carol Zander, Chris Loftus, and Anna Eckerdal. 2017. Student software designs at the undergraduate midpoint. In *22nd Conference on Innovation and Technology in Computer Science Education (ITiCSE 2017)*. ACM, 34–39.

[89] Errol Thompson and Kinshuk. 2011. The nature of an object-oriented program: how do practitioners understand the nature of what they are creating? *Computer Science Education* 21, 3 (2011), 269–287. https://doi.org/10.1080/08993408.2011.607010

[90] Neena Thota. 2014. Programming course design: phenomenographic approach to learning and teaching. In *2014 International Conference on Teaching and Learning in Computing and Engineering*. IEEE, 125–132.

[91] Laura Toma and Jan Vahrenhold. 2018. Self-efficacy, cognitive load, and emotional reactions in collaborative algorithms labs – a case study. In *13th International Computing Education Research Conference (ICER 2018)*. ACM, 1–10.

[92] Jodi Tutty, Judithe Sheard, and Chris Avram. 2008. Teaching in the current higher education environment: perceptions of IT academics. *Computer Science Education* 18, 3 (2008), 171–185. https://doi.org/10.1080/08993400802332423

[93] Jan Vahrenhold and Paul Wolfgang. 2014. Developing and validating test items for first-year computer science courses. *Computer Science Education* 24, 4 (2014), 304–333.

[94] L Vygotsky. 1987. *Zone of proximal development. Mind in society: the development of higher psychological processes, 52-91.* Cambridge, MA: Harvard University Press.

[95] Christopher Watson, Frederick WB Li, and Jamie L Godwin. 2013. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*. IEEE, 319–323.

[96] Susan Wiedenbeck. 2005. Factors affecting the success of non-majors in learning to program. In *First International Computing Education Research Workshop (ICER 2005)*. ACM, 13–24. https://doi.org/10.1145/1089786.1089788

[97] Timothy T Yuen and Kay A Robbins. 2014. A qualitative study of students' computational thinking skills in a data-driven computing class. *ACM Transactions on Computing Education (TOCE)* 14, 4, Article 27 (Dec. 2014), 19 pages.

[98] Carol Zander, Jonas Boustedt, Robert McCartney, Jan Erik Moström, Kate Sanders, and Lynda Thomas. 2009. Student transformations: are they computer scientists yet?. In *Fifth International Computing Education Research Workshop (ICER 2009)*. 129–140.

[99] Mark Zarb and Janet Hughes. 2015. Breaking the communication barrier: guidelines to aid communication within pair programming. *Computer Science Education* 25, 2 (2015), 120–151. https://doi.org/10.1080/08993408.2015.1033125

[100] Rui Zhi, Thomas W Price, Samiha Marwan, Alexandra Milliken, Tiffany Barnes, and Min Chi. 2019. Exploring the impact of worked examples in a novice programming environment. In *50th ACM Technical Symposium on Computer Science Education (SIGCSE 2019)*. ACM, 98–104.