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

Development and Use of Domain-Specific Learning Theories, Models and Instruments in Computing Education

Published in:
ACM Transactions on Computing Education

DOI:
10.1145/3530221

Published: 01/01/2023

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, & Sinclair, J. (2023). Development and Use of Domain-Specific Learning Theories, Models and Instruments in Computing Education. *ACM Transactions on Computing Education*, *23*(1), Article 6. https://doi.org/10.1145/3530221

Development and Use of Domain-specific Learning Theories, Models, and Instruments in Computing Education

LAURI MALMI, Aalto University, Finland
JUDY SHEARD, Monash University, Australia
PÄIVI KINNUNEN, University of Helsinki, Finland
SIMON, Unaffiliated, Australia
JANE SINCLAIR, University of Warwick, United Kingdom

Use of theory within a field of research provides the foundation for designing effective research programs and establishing a deeper understanding of the results obtained. This, together with the emergence of domain-specific theory, is often taken as an indicator of the maturity of any research area. This article explores the development and subsequent usage of domain-specific theories and theoretical constructs (TCs) in computing education research (CER). All TCs found in 878 papers published in three major CER publication venues over the period 2005–2020 were identified and assessed to determine the nature and purpose of the constructs found. We focused more closely on areas related to learning, studying, and progression, where our analysis found 80 new TCs that had been developed, based on multiple epistemological perspectives. Several existing frameworks were used to categorize the areas of CER focus in which TCs were found, the methodology by which they were developed, and the nature and purpose of the TCs. A citation analysis was undertaken, with 1,727 citing papers accessed to determine to what extent and in what ways TCs had been used and developed to inform subsequent work, also considering whether these aspects vary according to different focus areas within computing education. We noted which TCs were used most often and least often, and we present several brief case studies that demonstrate progressive development of domain-specific theory. The exploration provides insights into trends in theory development and suggests areas in which further work might be called for. Our findings indicate a general interest in the development of TCs during the period studied, and we show examples of how different approaches to theory development have been used. We present a framework suggesting how strategies for developing new TCs in CER might be structured and discuss the nature of theory development in relation to the field of CER.

CCS Concepts: • **Social and professional topics → Computing education;**

Additional Key Words and Phrases: Computing education, theory, theoretical construct, literature, research, instrument

Authors' addresses: L. Malmi, Aalto University, Espoo, Finland; email: lauri.malmi@aalto.fi; J. Sheard, Monash University, Melbourne, Australia; email: judy.sheard@monash.edu; P. Kinnunen, University of Helsinki, Helsinki, Finland; email: paivi.kinnunen@helsinki.fi; Simon, 2 The Ridge, Wadalba, NSW 2259, Australia; e mail: simon.unshod@gmail.com; J. Sinclair, University of Warwick, Warwick, United Kingdom; email: j.e.sinclair@warwick.ac.uk.

1 INTRODUCTION

Theories can guide researchers through all aspects of a research project, including selecting a phenomenon or problem to study, choosing the methodology, and interpreting the results [136, 137]. Theories are the fundamental tools of computing education researchers as we seek to cumulatively improve our understanding of teaching and learning computing. While the need for theory in research is clear, the definition of what constitutes theory and how theories are used is not, and what is regarded as a theory—or a valid theory—varies according to field of study.

As a new discipline, **computing education research (CER)** is gradually building its own identity [159, 174]. An important aspect of this process is the increasing adoption of appropriate theoretical frameworks and rigorous methodologies from the social sciences for the investigation of computing education. However, other disciplines can provide only general models and theories on learning and cognition, so it is essential that computing education researchers seek to develop new theories that address matters unique to computing education [107]. For example, learning to program is such a complex process that it cannot be fully explored without a deep understanding of programming concepts and processes, and therefore cannot be properly elucidated by reference only to theories from other fields of research. A concrete example of programming-specific theory is the **normalized programming state model (NPSM)** developed by Carter et al. [24], which "characterizes students' programming activity in terms of the dynamically-changing syntactic and semantic correctness of their programs"; they further explored how students' behavior in the model predicted their subsequent performance in the course. Another good example is the work by Xie et al., who presented a theory for instruction of programming skills, incorporating careful argumentation on separating, structuring, and sequencing programming skills [194]. In addition, there is much qualitative work that explores students' understandings of central computing concepts, such as class diagrams [16], code tracing strategies [45], interfaces [15], students' designs [177], and the nature of object-oriented programming [179], thus building micro-level theories on these topics.

The endeavor to develop domain-specific theories is also a sign of independence for a field of research, as defined by Fensham [43] in his detailed analysis of the related discipline of science education research. Computing education research clearly meets his structural criteria for a field, such as professorships in the area, dedicated research journals and conferences, and professional associations. CER also readily meets his intra-research criteria, such as creating scientific knowledge, asking distinctive research questions not addressed by other fields, applying research methodologies, building on previous scientific work, and having model and seminal publications. However, the criterion still in the emergent phase is conceptual and theoretical development, which is the focus of this article.

Conceptual and theoretical development has raised considerable interest in the field over the past few years. Several broad literature surveys have explored which theories have been used in the major CER publication venues and how they have been used. In earlier work [109], we studied research papers published between 2005 and 2011 in the **Computer Science Education journal (CSEd)**, **ACM Transactions on Computing Education (TOCE)**, and the **International Computing Education Research conference (ICER)**, exploring how they build on theoretical work

and in which disciplines this work began. Lishinski et al. complemented that work by analyzing ICER papers from 2012 to 2015, finding that "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" [93, p. 167].

Our own work went on to investigate papers published between 2005 and 2015 in the same three venues, but now focusing on theoretical work developed in the CER domain itself [107]. Subsequently, we examined a much wider span of venues, looking at literature that presented theoretical work specifically concerning emotions, affects, attitudes, beliefs, or self-efficacy in teaching and learning programming [108].

A recent ITiCSE working group paper by Szabo et al. [171] presents "a quantitative analysis of how learning theories are adapted in the computing education research communities," identifying a list of learning theories applied in research papers and influential papers within this pool of papers. Another working group paper, by Luxton-Reilly et al. [101], focuses on the literature of programming education, identifying—among many other results—papers that model student understanding in introductory programming.

In a recent special issue of Computer Science Education, Margulieux and Morrison [112] "selected papers that provide comprehensive literature reviews across several areas of interest, including K–12, higher education, sociocultural and cognitive factors. The literature reviews inform rigorous empirical studies and systematic literature reviews that build theory about novice programmers." Another recent work by Margulieux et al. [110] sets out to survey "commonly used variables and measurements in computing education and to compare them to best practices in measurement for human-subjects research." Finally, the *Cambridge Handbook for Computing Education Research* [44] focuses on theory in several chapters: Learning sciences for computing education [111], Cognitive sciences for computing education [143], and Motivation, attitudes, and dispositions [94].

This general interest is well in line with the practice of several major CER publication venues in emphasizing the role of theories in their instructions for authors. For example, from the very beginning of the conference series in 2005, ICER has solicited papers with *"A clear theoretical basis, building on existing literature in computing education, computer science, and other related disciplines,"* and the Koli Calling conference has for several years called for *"theoretical research papers focused on deriving a better understanding of the process of teaching/learning computing or of conducting research in computing education."*

The Computer Science Education journal calls for *"submissions that are situated in the computing education research literature and draw on diverse theories and research methods, including those that borrow from allied fields such as educational psychology, cognitive science, and the learning sciences,"* and ACM TOCE advises that *"paper authors are strongly encouraged to frame their research in terms of one or more learning theories. This can be accomplished, for example, by motivating research questions or hypotheses with the help of learning theory, or by interpreting results in terms of learning theory."*

This interest in theories is understandable because, as a multidisciplinary field, CER draws heavily on theory from other disciplines, such as psychology and education [111, 143, 171]. However, such an approach has also been criticized. Nelson and Ko [123] argue that our field must have space for developing new educational designs even though they may lack a rigorous theoretical framework. As our understanding of how students learn computing is still far from complete, there must be room for designing and exploring new pedagogical solutions even when we have no theory to guide us. Nelson and Ko emphasize that the field should focus more on developing domain-specific theories than on applying more general theories of learning.

Based on these examples, as well as on general interest in theories in CER, we conclude that the case is clear for CER to have its own approaches to the development of theory. This can be implemented in several ways. First, existing theoretical constructs from other disciplines could be contextualized in computing education. Examples include theoretically grounded instruments, such as the cognitive load scale for introductory computer science [119] and various versions of the computer programming self-efficacy scale [77, 135, 181]. Second, current computing learning environments can collect rich datasets. Analysis of this data can help us to build models of students' learning or study processes that identify factors influencing the process and results, and that can be used to predict results. A good example of such work is the **normalized programming state model (NPSM)**, which describes students' process in compiling and running their programs [24]. Third, such models can be integrated with theories from the social sciences to more richly explain the learning process, as with the work that combines the NPSM with social cognitive theory to build a new social NPSM, leading to improved prediction results. Another example is the work by Parkinson and Cutts [127], which looks into the relationship between spatial skills and computing ability.

In this article, we seek to find a deep insight into the nature and purpose of domain-specific theories in CER and how they are being used in our field. While previous surveys [93, 107, 109] have focused mainly on identifying *what* theories have been developed, we aim to look more deeply into their *nature*. We categorize domain-specific theories based on their broad purpose; for example, to analyze, to explain, or to predict phenomena. Moreover, we explore how the theories have been used to inform further research and how these aspects vary in different focus areas within computing education. We hope that this exploration will give us insights into any trends in theory development and suggest where more work might be appropriate. Moreover, by a closer investigation of cases where CER theories are combined with one another or with theories from disciplines such as the social sciences, we aim to distinguish how these different combinations can enrich our understanding of teaching and learning computing. Finally, we reveal good examples of ways in which theories have been developed over time, helping with the accumulation of our understanding of computing-education-related phenomena.

In the next section, we discuss theories and their role in research on a general level. We proceed to present our research questions, data collection, and analysis methods in Sections 3 and 4. In Section 5, we present results from our literature survey, focusing on quantitative findings of theories, their nature, and how they have been used. Thereafter, in Section 6, we give a set of examples of different types of theory that have been developed in CER, followed by general discussion of our findings and theory development strategies in Section 7. We conclude with a brief summary and suggestions for researchers in CER.

2 THEORIES IN RESEARCH

In this section, we first discuss theories in education and social science from the perspective of the philosophy of social science, arguing that theories in social science have a different role from those in natural science and that models are important forms of describing complex phenomena. Thereafter, we take a closer look at the characteristics of theories in the social sciences and consider the implications for CER as a multi-disciplinary field that addresses a wide variety of phenomena.

2.1 Theories in Educational and Social Science Research

The need for theory in educational research has been justified by several authors over the decades. As early as the mid-1970s, Suppes [169] argued for the relevance of theory by stating that we need theory to guide us to perceive what is important, recognizing the complexity of seemingly simple matters, and to avoid the triviality of bare empiricism. Discussion of the definition and function of theory in educational research has continued more recently, with a collection of papers on the role

of theory [137] highlighting that we are dealing with a manifold issue that encompasses several possible approaches to the topic. Educational science, and the social sciences in general, are based on multiple ontological and epistemological beliefs about the nature of knowledge and how it is possible to gain new knowledge. To further complicate the setting, epistemological beliefs are also affected by trends in the politics of science and education.

If we dig more deeply here, we encounter the debate of the nature and goals of scientific inquiry in general, a topic discussed at length in the philosophy of social science. While thorough discussion of these debates is well beyond the scope of this article, we briefly summarize some main aspects of it, as it helps us to explain how we interpret "theory" in the context of CER.

The major question addressed in the philosophy of social science, as formulated by Rosenberg [145, p. 7], is "whether human action can be explained in the way that natural science explains phenomena in its domain." If our answer is yes, a follow-up question asks why the explanations concerning human actions are less precise and give rise to weaker predictions than those in natural science. However, if we disagree, then we should explain why we need a different approach to scientifically explain human action and what approach might be better suited to this task.

Risjord [140] asks: "Can human action be causally explained? Anti-naturalists argue that it cannot because humans act for reasons, and reasons are not causes." Moreover, causal analysis requires the existence of laws, but it is not clear that there are laws in the social world in the same sense as in the realm of the natural sciences.

Machamer and Silberstein [102, ch. 14], citing Durkheim [39] and Weber [189], summarize these debates as

(1) *Naturalism vs. antinaturalism*: "do the social sciences proceed using the same methods and criteria of adequacy as the natural sciences?"
(2) *Interpretism vs. causalism*: does explaining social phenomena require a fundamentally different interpretation of meanings, thus making them different from the natural sciences; or are (social) phenomena susceptible to ordinary causal explanations?
(3) *Instrumentalism vs. realism*: can/should/must models in the social sciences "be taken as literal descriptions of social reality rather than as illuminating stories with some other less direct connection to reality?"
(4) *Holism vs. individualism*: can/must the social sciences explain in terms of irreducible social entities or must all adequate explanations be given in terms of individuals?

These are fundamental questions for scientific inquiry. In the natural sciences, the goal of inquiry is generally seeking and identifying regularities and their underlying laws, because causal knowledge (and the technological control that relies on it) requires regularities. The challenge in the social sciences is that due to the fierce complexity of human beings, social science is much harder than natural science. When it comes to investigating human behavior, the presence of the multitude of intervening variables makes it highly difficult to test regularities.[1] First, it is rarely possible to isolate single aspects for careful observation and scrutiny. Second, observation and investigation are likely to interfere with the phenomenon; for example, students' behavior can change if they know they are being observed. Third, no observation or investigation can truly access the actual thoughts, conceptions, attitudes, or feelings of human participants. If study participants are asked to verbalize these, then the result is likely to be incomplete due to forgetting details or lacking the capability to be accurate. The results can also be biased due to intentionally mentioning only some aspects, or possibly even deliberately giving false or distorted data.

[1]We do not seek to devalue the huge technical complexity of measurements in, for example, subatomic particle physics experiments.

Consequently, repeating an investigation with the same individuals or cohorts is likely to give somewhat different results. Finally, on an individual level, the basic tenet is that individuals are different, and therefore that results from any two people are likely to be different. All of these challenges also have ethical implications for research.

While the natural sciences seek to identify causal theories that enable us to predict and control, much of the work in the social sciences seeks to "explain behavior by rendering it meaningful or intelligible" [146, p. 21]. The goal is to uncover its meaning, or significance, by interpreting what people do. Moreover, the interpretation of human behavior, in this view, is not fundamentally causal. Rosenberg concludes: "Influential social scientists since Max Weber have argued that theories about human behavior, human action, and human institutions need to uncover both causal laws and interpretative meanings, and that this dual requirement is what distinguishes social from natural sciences" [146, p. 24]. This certainly also applies to CER.

These issues relate directly to the goals of theoretical discourse. If we reject the premise of the natural sciences that progress in theories is measured in terms of their capability to make more accurate predictions and causal explanations, then we can conclude that theories in the social sciences have a different goal of explaining and interpreting reality, and their progress entails extending the variety of real settings addressed by theories. We can build models that describe complex human phenomena in different ways, some simpler, some richer in detail. These models seek not just to describe the external behavior that can be observed, but also to explain the human actions by identifying the beliefs and desires that lead to them. Williamson [193, p. 159] formulates the role of models as follows: "When a system resists direct study, because it is so complex or hard to observe, model-building constitutes a key fall-back strategy. Studying a model often yields insight into the phenomena it models." He subsequently notes that it is hard to avoid the need for model-building in cases where the complex, messy nature of the subject matter tends to prevent building informative and exceptionless universal generalizations. Such complexity and mess is the nature of the human world.

Acknowledging these premises, our aim is not to propose one overarching definition of theory in CER, but rather to nurture multiple understandings of theory and models.

2.2 Different Natures of Theories in Education

Educational phenomena are rich and varied, and researchers have adopted different approaches to studying and making sense of educational issues. Bredo [19] classifies epistemologies of educational research into three main families, helping us to make sense of the diversity of theories that have been introduced in educational research and more specifically in CER. Research in the first family seeks to establish general laws of behavior. Research in the second family focuses on understanding and describing the rich and ever-evolving phenomena of education. Research in the third family is interested in dialectical and/or transactional relationships, aiming, for example, to empower actors or to develop existing situations. Standing aside from the three families is pragmatism, which approaches the notion of knowledge by focusing on how useful it is in guiding action.

Another way to grasp the differences between the theories found in CER derives from Gregor [49], whose discussion of the structural nature of theory in information systems includes a taxonomy that differentiates theories based on their main goal:

(1) *Analysis* theory states "what is": phenomena are analyzed and described but no causal relationships or predictions are made;
(2) *Explanation* theory states "what is, how, why, when, and where," explaining some aspect of the phenomena but not aiming at prediction;
(3) *Prediction* theory states "what is and what will be," offering predictions;

(4) *Explanation & prediction* theory enables both causal explanations and predictions;
(5) *Design & action* theory states "how to do something."

These five types of theory compare well with Bredo's families of philosophies of educational research [19]. Gregor's Prediction (3) and Explanation & prediction (4) theories aim at finding general laws of behavior and relate to Bredo's Family 1, general laws, and consequent predictions. Gregor's Analysis (1) and Explanation (2) have much in common with the basic tenets of Bredo's Family 2, describing and understanding rich and ever-evolving phenomena. Finally, Gregor's Design & action (5) has flavors of both pragmatism and Bredo's Family 3, underlying many studies that aim at improvement and/or empowerment.

Gregor also suggests that theories of different types feed into one another [49]; for example, analysis theories are needed when one is developing other types of theory. Therefore, Gregor's taxonomy of theories also represents possible paths, albeit not necessarily linear, through which theories can evolve and mature to become more versatile and more powerful. This idea of theories evolving through stages opens up an opportunity for the research community to explore the stage that has been reached by the collective understanding of the phenomena under investigation.

2.3 Perspectives of Theories in CER as a Multidisciplinary Field

CER is not a pure subarea of educational research, addressing various aspects of teaching and learning computing using only theories from education and other social sciences. Instead, CER is also a subarea of computing in the sense that CER builds novel artifacts, both software and hardware, to support learning using theories, methods, and approaches from computing proper. Moreover, computing itself combines traditions from mathematical research, scientific research, and engineering [175], each having different interpretation and use of theories.

In the context of this article, we can ignore theories in the sense of mathematical proving of theorems. While theorems and proofs are addressed as the domain subject of learning in the context of, for example, theoretical computer science and algorithms research, they are hardly ever used in addressing the processes involved in teaching and learning computing. The case is different when we consider the engineering tradition in computing, where the focus is on designing new artefacts. In CER, design work is essential in the context of building new educational software/hardware, an area often called tools research. Morever, CER also designs novel teaching and learning approaches and environments. This design work may be guided by existing theories, mainly from the social sciences, but this is not necessary, as here the primary driving force is exploration and improvement, which cannot be strictly tied to the use of existing theories [123].

Most work in CER, however, follows the scientific tradition of both computing and the social sciences in the sense that the core of CER is empirical work—and empirical work is heavily intertwined with theories. Theories can guide research designs; they can be used to interpret findings or predict new findings; empirical results can be used to support theories or discount them; empirical results can be used to extend existing theories or build wholly new theories. In CER, most of the use cases of theories are related to theories from the social sciences. However, CER also builds its domain-specific theories, which are the focus of this article.

Since the contexts and phenomena dealt with by CER are varied, there is a need for different research methods in addition to multiple theories. Qualitative research can help us to identify different conceptions, structures, and processes within the individual or cohort being investigated, consequently allowing us to describe the phenomenon in a rich way, with an overall goal of building an initial model of its internal working. In the terms of Gregor [49], such models are typically Analysis or Explanation types of theory. Some examples include phenomenographical outcome spaces describing students' different levels of understanding of class diagrams [16] or Java

interfaces [15]. More elaborated cases include the work by Kinnunen and Simon [75, 76], which describes the different phases and emotional experiences that students have when facing difficulties during a CS1 course.

The approaches mentioned above can also help us to formulate more accurate hypotheses about the phenomenon, which can then be explored using quantitative research to identify the statistical characteristics of factors and their relationships, thus enriching Explanation or Prediction theories. Depending on the setting, research can also support identification of the underlying causal relationships. For example, Lishinski et al. [96] explored the role of self-efficacy in programming, and Stout and Blaney [168] investigated how intellectual belonging to CS influences persistence to study CS. Together, the qualitative and quantitative approaches, despite their methodological differences, aid us to address the "how and why" questions, thus building our understanding of the phenomenon on an abstract level that transcends the individual observations (Explanation & Prediction). They can also guide the formulation of theoretically grounded practices and pedagogies (Design & Action), such as the 3-motivator theory by Nikula et al. [124].

In summary, empirical research can distill information from a complex phenomenon to explain what is happening and how and why it happens. The answers to these questions help us to formulate a description of the phenomenon or system, which summarizes some of its working at a more general and abstract level, thus formulating models or theories—although the quality and scope of the evidence and argumentation vary with different types of theory. Only in this way can we overcome the morass of detail that would otherwise overwhelm us.

Finally, we note that often the same phenomenon or system can be addressed in the light of several different theories. Social contexts are complex and can be investigated from many different angles. Reeves et al. [136, p. 631] formulate this as follows: "Theories give researchers different 'lenses' through which to look at complicated problems and social issues, focusing their attention on different aspects of the data and providing a framework within which to conduct their analysis." In exploring a pedagogical setting where students are learning a particular computing concept, one lens might focus on their understandings of the concept; another might focus on their motivation to study the concept and how this affects their engagement with the learning content; and a third lens might focus on the interactions among students in the pedagogical context where they are studying the concept. All of these address the same phenomenon, but together they can provide a much richer and deeper understanding of the underlying process whereby students study and learn the concept in question.

2.4 Scope of Theories and Computing Education

Theories can be categorized in several ways, depending on how widely they can be applied. Reeves et al. [136] split theories according to their breadth of coverage: *micro-level theories*, *mid-range theories*, and *grand theories*. In computing education contexts, this split might be interpreted as follows: Micro-level theories address a single context, such as teaching/learning some specific concepts in a specific pedagogical setting. Most domain-specific theories in CER are on this level. Mid-range theories could address wider systems; for example, exploring how different factors influence recruitment to study computer science, taking into account cultural and contextual variation. CER has no grand theories such as, for example, social constructivism in educational science.

However, theories focus on different aspects and concern different target groups. Adults study and learn differently from children, and children's learning depends on their age group [62]. This suggests that theories could be categorized based on their intended target group. In computing education, this dimension reaches from preschool learning to university education and adult education. Furthermore, evidence for a theory can be collected from a single class, from a university, from several institutions, or even multinationally. At first glance, it could seem that theories based

on multinational data collection would be "stronger" than theories with more narrowly scoped data. While this might be true, the challenge is that broad data collection could easily overlook institutional and cultural differences that introduce intervening variables that we are not aware of. The same challenge also applies to replication studies, although they might have more opportunity to control the settings.

Another dimension of focus concerns the domain subject matter that the theory addresses. There is no comprehensive theory of learning computing, and it seems unlikely that there ever will be. Rather, theories address different topics in computing, such as learning programming [41], computational thinking [156], learning theoretical computer science, and so on. On a more refined level, theories can focus on single concepts, such as learning recursion [87] or interfaces [15].

A different but related dimension concerns the focus area of investigation. In an earlier survey of domain-specific theories in CER [107], we identified 11 distinct areas of focus for theories, which address the following types of questions:

Learning/understanding: How do students learn or understand one or more computing-specific topics or concepts?
Learning behavior/strategies: How do students act in their learning or studying process?
Teaching/pedagogical content knowledge: How do teachers act in their teaching, and what knowledge or skills do they use to guide their teaching?
Assessment/self-assessment: How is learning of computing topics being assessed?
Performance/progression/retention: How well do students learn computing topics, or how do they progress in their studies?
Errors/misconceptions: How can students' errors and misconceptions be characterized?
Contents/curriculum/learning goals: What are the learning goals at course or curriculum level, and how are these goals and topical content defined?
Emotions/attitudes/beliefs/self-efficacy: What kinds of emotion, attitude, or belief do students associate with learning computing, or how does their self-efficacy in learning computing influence their work?
Perceptions of computer science/computing: How do students perceive computing or computer science when studying it?
Study choice/orientation: What factors relate to students' decisions to study computing?
Theories of computing education research: What aspects characterize CER as a research field, or how is it carried out?

This categorization was data-driven, and the questions above are generic examples of topics that theories in these areas address. It is clear that other classifications within this dimension are possible.

3 THEORIES, THEORETICAL CONSTRUCTS, AND RESEARCH QUESTIONS

The discussion in Section 2 reveals the richness of topics and points of view that theories can address in CER. As a major component of CER can be considered a social science, we therefore build on Rosenberg's conclusion that the social sciences seek to uncover both causal laws and interpretive meanings [146, p. 24]. These can be presented in a multitude of ways with different terminologies, such as "theory," "model," or "theoretical framework." Models, especially, can be used to present a complex phenomenon at varying levels of abstraction, using mathematical, diagrammatic, or verbal presentations. However, interpretive meanings are often presented as qualitative categorizations, where the categories form some kind of structure that explains their differences.

Due to the diversity of terminology and presentation formats in the literature, some broad definition for theories is needed. Szabo et al. discuss this issue in their review of learning theories

in CER [171] and choose to use "an inclusive definition of theory as a generalisation, abstraction, explanation or prediction of a phenomenon . . . where the phenomenon under study is learning" [p. 92]. However, we have decided to continue to use the broad term ***theoretical construct (TC)*** that we introduced in our earlier surveys of the literature [107, 108]. TCs thus capture theories, models, frameworks, and even theory-based instruments (see below). We use the following guidelines for identifying TCs:

- Various statistical models such as regression models, path models, factor analyses, structural equation models, and clustering, because these are typically generated from collected empirical data with the purpose of explaining the relationships between identified concepts.
- Qualitative data-driven categorizations that build higher-level abstract descriptions of the data; for example, grounded theories or phenomenographical outcome spaces. Simple lists of qualitative categories are excluded, unless the paper also includes a clear discussion of structural relationships among the categories.
- Other types of explanatory models, such as typologies and taxonomies.
- In some cases, we also include figures and formulas, when they clearly build an abstract description of the data using logical argumentation or are derived from other theoretical frameworks and adapted to address the relevant domain.

Finally, we also decided to include *validated instruments* for measuring particular theory-based concepts. These can be valuable tools for building new theoretical constructs, and our goal is to explore not only the publications in which novel theoretical constructs are presented but also how they are used in subsequent theoretical development in papers that cite the original papers. Validated instruments embody TCs in a concrete way, and by linking the underlying theory to specific measurable responses they contribute to our understanding of the underlying concepts. Not only do validated instruments meet the requirement of explanatory capability, but they can in practice be easily adopted by other researchers, leading to wider reuse.

Our focus in this research is specifically on domain-specific TCs that have been developed within CER. We explore the nature of the questions that they seek to address and how they are distributed within different focus areas (as listed at the end of Section 2.4) and different educational contexts (such as K–12 computing, introductory programming, and capstone projects). Moreover, we survey how these constructs have been used to inform further research and observe how this is related to the nature and scope of the construct. This offers guidance as to where further theoretical development might be needed and what form that development might best take.

More specifically, we address the following research questions:

RQ1. What domain-specific theoretical constructs have been developed in particular areas of CER?
We investigate constructs that have been introduced in certain specific research venues and that pertain to a number of distinct areas of focus (see the end of Section 2.4 for the list of areas). We look for any emerging trends within venues and within focus areas.

RQ2. How were the theoretical constructs developed?
We explore and classify the methodological approaches used to derive the constructs based on some combination of empirical work, the use, adaptation, or extension of existing theories, and argumentation.

RQ3. For what purposes have domain-specific theoretical constructs been developed in CER?
We classify the theoretical constructs in a number of ways: their nature, their scope, and the educational level to which they have been applied. We examine the spread of constructs across the venues and the areas of focus that we have chosen for this study.

RQ4. How have these theoretical constructs been actually used in further research?
We examine the papers that cite the papers in which the theoretical constructs were presented, determining the nature and purpose of the citations. Do the papers cite the paper for the theoretical construct itself, or for some other aspect of the source paper, such as its findings? If they cite the source paper for the construct, do they go on to make some use of it in their own work, in which case, what sort of use?

RQ5. What kinds of theoretical constructs have been widely used and what kinds of constructs have raised little attention in papers citing the original paper?
We look at constructs that show much or little evidence of having been used in subsequent research, and see whether we can postulate a likely reason for this.

In the next section, we give further information on how the TC guidelines were applied and how the analysis was carried out.

4 METHOD

To answer the research questions, we collected and analyzed data on the nature and usage of theoretical constructs found in CER publications. It was necessary to define a clear scope for the data collection that, while resulting in as complete a picture of construct use as possible, was feasible to implement. While in theory there are many conferences and journals in which CER might appear, our prior work [108] has indicated that in practice, the venues publishing the highest numbers of relevant papers are the **ACM International Computing Education Research Conference (ICER)** and the journals **Computer Science Education (CSEd)** and **Transactions on Computing Education (TOCE)**. Further argument is that these three venues publish long papers, which allow more space for thorough discussion on literature and theory than can be included in shorter conference papers. We therefore restricted our consideration to these publications. With that restriction, we know that we have not found every TC used in the context of CER. However, the result of considering these major venues is an extensive and rich dataset that covers a high proportion of the relevant work and is likely to be broadly indicative of the topics and trends of the evolving research area.

The ICER conference was first held in 2005 in response to the growing interest in computing education as an area of research. We took 2005 as the starting point for the data collection, with all three main publications operational from that point, and a clear growth in computing education research apparent.

A further decision to limit scope was taken relating to the type of article we would consider. Some publications include editorials, short papers, posters, and so on, but in practice these seldom present new TCs. Many report early stages of work that, if successful, is highly likely to be submitted later as a full article. We therefore excluded such publications on the basis that this was unlikely to result in many theoretical constructs being missed.

The resulting scope of the data collection was that we considered all full articles appearing in ICER, CSEd, and TOCE in the 16 years from 2005 to 2020. This gave a total of 878 papers: 330 from ICER, 304 from TOCE, and 244 from CSEd. We note that each of these venues has a thorough process of peer review to which all full papers will have been subjected.

To address RQ1, we identified all full research papers within our dataset that present a domain-specific theoretical construct; we call these papers *source papers*. While the above-mentioned main categories were our guidelines, there were numerous cases where further decision-making was needed. Due to space limitations, we cannot list comprehensive inclusion/exclusion criteria for all different cases. However, the following list provides the most relevant cases with some examples of exclusion:

(1) For practical reasons, we did not include single hypothesis testing between two variables, because this would capture so many results from quantitative research that our pool of TCs would be overwhelming, making this survey more or less meaningless.
(2) We also excluded hypothetical models where the researchers have not found empirical support. An example is Guzdial's summary of his team's long-time research on impacts of media computation on students' retention [50], where he proposes a hypothetical model of the impacts. However, he notes explicitly that they lack evidence for certain hypothetical impacts, because they have not explored them.
(3) Statistical models that were based on analyzing data from very specialized target groups or research settings were excluded, as we deemed their external validity too low.
(4) Methodological contributions, where the paper focus was on the development process of a model rather than using the model to discuss results (e.g., Reference [91]), were excluded.
(5) In qualitative research, many papers apply grounded theory methods. However, if the results include only a list of categories identified during open/axial coding with example quotes, or only partial results of focusing on some categories (e.g., Reference [133]), then we decided to exclude the potential TC.
(6) Similarly, many papers present content analysis results without clearly explaining how the categories relate to one another to form a bigger whole with an argued structure, and correspondingly miss our criteria for inclusion.
(7) Numerical groupings that were excluded are those that are used without explanation or justification, perhaps for convenience of gathering user feedback but with no clear basis. For example, respondents to "What percentage of the module videos did you watch?" might be asked to tick one of 0–20%, 21%–40%, and so on. Such groupings do not meet our criteria for inclusion.
(8) We excluded pedagogical models and frameworks and design principles for learning resources, e.g., References [42, 100], when their main purpose was not to explain empirical findings but to provide guidelines for teachers, even though these may have been well grounded in theories. However, we identified a small number of papers that present pedagogical frameworks that are grounded in existing theories and that include rigorous empirical evaluation to validate the pedagogy, e.g., References [7, 124, 194]. These were included.

To improve the trustworthiness of the process, each paper was independently examined by two members of the research team, with any differences of opinion resolved by discussion to reach consensus. No formal measure of **inter-rater reliability (IRR)** was made, but it was clear from the discussion stage that agreement was very high and that differences were generally due to simple oversights rather than to fundamental disagreement. While it might appear unusual not to measure IRR when classifying, we make the point that the intent of our work is to present the big picture of CER in particular focus areas and that any irregularities in the classification are unlikely to have any perceptible impact on that big picture.

For each domain-specific TC identified, more information was recorded by two or three researchers, always seeking consensus for the findings. This additional information includes a brief description of the theory found, its focus area in CER (RQ1), the research methodology used to develop it, and whether it builds on or combines previous theory (RQ2).

As a basis for the focus area categorization, we used the data-driven classification of CER areas of focus that we developed in earlier work [107], which comprises the 11 broad categories presented in Section 2.4. While we recorded TCs in all 11 focus areas, we constrained the scope of this study by analyzing only those categories that relate to student learning, studying, and progression, rather than, for example, categories to do with learning goals or content knowledge. The resulting

categories used in our classification are *learning/understanding*, *emotions/attitudes/beliefs/self-efficacy*, *learning behavior/strategies*, *performance/progression/retention*, *assessment/self-assessment*, and *errors/misconceptions*.

For methodological categorization, we listed the major qualitative and quantitative approaches used. When the TC development was based on using, modifying, or extending some existing TC, either from CER or from other disciplines, this was listed separately. In cases where development relied on substantial use of literature and/or argumentation, this was also noted. We acknowledge that any research includes literature and argumentation, but when developing TCs these facets sometimes play a dominant role.

As a starting point for investigating the nature of the domain-specific TCs (RQ3), we categorized them according to Gregor's taxonomy of theories [49]. Derived from the related field of information systems, this meta-theoretical framework is highly relevant to the current investigation and (as described in Section 2.2) provides a high-level mapping to the broad categories of *analysis*, *explanation*, *prediction*, *explanation & prediction*, and *design & action*. Theories with these different purposes feed into one another; for example, analysis theories are often needed when other types of theory are being developed. Thus, Gregor's taxonomy of theories also represents possible paths, albeit not necessarily linear, through which theories can evolve and mature to become more versatile and more powerful. This idea of theories evolving through stages opens up an opportunity for the research community to explore the stage that has been reached by the collective understanding of the phenomena under investigation. As with the earlier classification step, two researchers assessed each paper independently, with subsequent discussion and resolution of any papers on which they were not in agreement.

For each source paper found in the previous step, a search was performed to identify papers that have cited it (RQ4). These *citing papers* form the dataset for the task of determining which TCs have been reused and in what ways they have been further developed. Google Scholar was used to identify papers and dissertations that had cited the source paper between its date of publication and the end of 2020. Google Scholar does not list all papers that cite a source paper, and lists some papers that do not cite it, but it appears to provide the most comprehensive listing of citing papers.

Papers were excluded from the list of citing papers for analysis if they are not accessible free of charge to members of our research team, are not written in English, or are duplicates or reprints of other papers in the list. We also excluded published abstracts, for example for posters or panel sessions. If a paper had more than about 30 citing papers, then we made the pragmatic decision to select the 30 or so most recent ones, because it was not practical to examine several hundred citing papers for a single source paper. The choice of 30 is explained in Section 5.4. The citing papers were not always from peer reviewed sources, but non-reviewed papers were not excluded, as they can still help provide a picture of how and where TCs are being referenced and used.

Each remaining citing paper was then assessed to determine in what ways the source paper had been used. The framework adopted for this assessment was a slight variation of one that we developed earlier [108]. As shown in Table 1, this framework categorizes the nature of TC usage according to 13 descriptors structured into four groups of related items. This allows us to address RQ4 by determining, for example, which citing papers merely refer to the source, and which papers actually use the theoretical construct in some way, building on it, using it as a data collection instrument, validating it, and so on. The difference from the original framework [108] is that we have combined all of the "description" levels into a single category, D. This is because our goal is to investigate how theory is used and developed, and it is not necessary to distinguish between different levels of reference where no other type of usage is observed. Indeed, in many cases where a source paper is referenced but no further use is made of the TC, the citing paper does not mention the TC at all.

Table 1. Categories of Use of TCs

Code	Description
D	Paper cited, some aspect described, discussed, and/or critiqued
A1	Used as a framework to scope a study
A2	Used to develop a data collection instrument
A3	Used as a framework for data analysis
A4	Used to predict results of a study
A5	Used to interpret/compare/explain results of a study
A6	Used to design a new pedagogical method
A7	Used as an instrument in a study
C1	Modified and/or extended existing construct for use in a new context
C2	Developed new construct from existing construct and empirical work
C3	Developed new construct from existing construct and argumentation
V1	Used in a test to improve or discount an existing/new construct
V2	Tested the construct in a new context

D – Description; A – Application; C – Construction; V – Validation (adapted from Malmi et al. [108]).

The datasets developed in this way provide the basis for further analysis to determine which types of theory have been most commonly used and which have been least used (RQ5). Examining this data for each focus area of CER allows us to compare and contrast the types of theory used and how they are being used in different focus areas. Differences in the profile of usage might indicate different approaches or levels of maturity between the focus areas. The quantitative data provide a map to actual usage and support analysis of the occurrence and development of theory use.

The current work builds on our previous analysis, extending it in two ways. It complements existing findings [107] by investigating a larger dataset consisting of all papers published in TOCE, CSEd, and ICER in the years 2005–2020, incorporating later years than have previously been studied. This allows a greater period of time over which to view reuse and development and can indicate whether the use of TCs is changing over time. The work also considers a broader range of focus areas, covering the broad subarea of CER that addresses students' learning, studying, assessment, and performance as well as experiences.

5 RESULTS

In this section, we present our findings for each research question in turn.

5.1 [RQ1] What Domain-specific Theoretical Constructs Have Been Developed in Specific Areas of CER?

Our search for new **theoretical constructs (TCs)** developed in the computing education area found 123 TCs reported in 121 papers from ICER, CSEd, and TOCE over the 16 years from 2005 to 2020. Brief descriptions of each TC are presented in Appendices A and B, the latter being specifically for instruments. Two papers [10, 35] report both a non-instrument TC and an instrument, and these are described separately in the respective tables.

The TCs were classified into 11 areas of focus. The numbers of papers for each area of focus are presented in Table 2. In the work reported in this article, we concentrate on TCs developed in six areas of focus: *assessment/self-assessment*, *emotions/attitudes/beliefs/self-efficacy*, *errors/misconceptions*, *learning/understanding*, *learning behavior/strategies*, and *performance/progression/retention*, as marked with asterisks in Table 2. We found 79 papers reporting 80 new TCs in these areas, with one paper in the area of *performance/progression/retention*

Table 2. Counts of Source Papers Reporting a New Theoretical Construct for Each Area of Focus

Area of focus	Count	Percentage
learning/understanding*	25	21%
emotions/attitudes/beliefs/self-efficacy*	22	18%
study choice/orientation	15	12%
performance/progression/retention*	12	10%
learning behavior/strategies*	10	8%
perceptions of computer science/computing	10	8%
teaching/pedagogical content knowledge	9	7%
assessment/self-assessment*	6	5%
content/curriculum/learning goals	5	4%
errors/misconceptions*	4	3%
computing education research	3	2%
Total	121	

The focus areas analyzed in detail in this article are marked with asterisks.

reporting two TCs. Not included in our further analysis are papers reporting TCs with a focus on *computing education research*, *content/curriculum/learning goals*, *perceptions of computer science/computing*, *study choice/orientation*, or *teaching/pedagogical content knowledge*.

Table 3 presents the numbers of papers from each year for each venue across all focus areas and separate totals for the six focus areas analyzed in this article. Overall, nearly half of the papers reporting a TC (47%) were from ICER. For the first five years of the data collection no TCs were reported in TOCE, which appears to align with a change in the journal's focus after this time. The largest number of papers for any venue in one year was for TOCE, which in 2020 had 10 papers presenting some TC. However, 9 of those 10 TCs are in the areas of study choice/orientation, perceptions of computer science/computing, content/curriculum/learning goals, and teaching/pedagogical content knowledge, and therefore fall out of scope of the more detailed analysis that we present below. It is interesting that this high concentration of TCs was not the result of a special issue, as TOCE did not have any special issues in 2020.

Table 4 lists the source papers for each of the six areas of focus chosen for further analysis in this article. The greatest number of TCs are in the learning/understanding focus area, closely followed by emotions/attitudes/beliefs/self-efficacy. The smallest groups are assessment/self-assessment and errors/misconceptions.

5.2 [RQ2] How Were the Theoretical Constructs Developed?

Our analysis identified many different approaches to developing the TCs. Typically a combination of methods was used. In 35% of the cases, the development of the TC involved the use of an existing theory, model, or instrument. Table 5 lists the source papers for each focus area that used such a construct in developing the new construct.

Quantitative methods were used in the development of 65% of the TCs, with regression the most used technique (14%). Qualitative methods were used in 25% of the cases, with the most common methods being phenomenography (13%) and grounded theory (8%). Literature analysis and argumentation, often in combination, were used in developing 28% of the TCs. While all papers include some argumentation, we listed argumentation only for cases where it played a substantial role in developing the TC. Table 6 lists the source papers that use a method or approach from

Table 3. Counts of Papers Reporting the Development of a Theoretical Construct (TC) in ICER, CSEd, and TOCE from 2005 to 2020 for All Areas of Focus and for the Six Areas of Focus Analyzed in the Work Reported in this Article

Year	Focus areas used in study (6)			All focus areas (11)			
	ICER	CSEd	TOCE	ICER	CSEd	TOCE	Total
2005	4	1	0	5	1	0	6
2006	3	2	0	3	2	0	5
2007	1	0	0	2	2	0	4
2008	2	0	0	2	1	0	3
2009	1	2	0	2	2	0	4
2010	2	1	0	4	2	1	7
2011	3	1	1	5	2	3	10
2012	1	3	0	3	3	1	7
2013	4	2	0	5	3	1	9
2014	4	1	1	5	2	2	9
2015	1	2	1	1	2	1	4
2016	2	1	3	4	1	3	8
2017	3	4	1	3	4	1	8
2018	5	0	1	5	0	2	8
2019	3	5	2	4	6	6	16
2020	4	0	1	4	2	10	16
Total	43	25	11	57	35	31	121

Table 4. Source Papers for the Six Areas of Focus Analyzed in this Work

Area of focus	Count	Source papers
learning/understanding	25	[11, 15, 22, 38, 41, 53, 61, 71, 87, 98, 119, 127, 138, 139, 142, 149, 150, 153, 156, 165, 177, 179, 182, 192, 196]
emotions/attitudes/beliefs/self-efficacy	22	[7, 13, 27, 28, 33, 34, 70, 75, 76, 78, 79, 81, 95, 96, 103, 115, 124, 144, 155, 167, 168, 199]
performance/progression/retention	12	[9, 10, 21, 25, 73, 90, 104, 122, 131, 134, 172, 180]
learning behavior/strategies	10	[16, 18, 24, 45, 66, 68, 97, 113, 163, 198]
assessment/self-assessment	6	[4, 37, 82, 132, 157, 173]
errors/misconceptions	4	[69, 114, 125, 184]
Total	**79**	

these areas for the development of the new construct in each focus area. We present many of these approaches in more detail with concrete examples in Section 6.

Most development was done with data collected in a single course (48%), or, less frequently, across a program (10%). We found a number of cases where the data was collected from multiple institutions and sometimes from multiple countries. However, it was often not clear from the reporting whether the multi-institutional or multinational perspectives were central or even relevant to the goal of the development of the TC.

5.3 [RQ3] For What Purposes Have Domain-specific Theoretical Constructs Been Developed in CER?

We classified the TCs in the six focus areas according to their main purpose, following Gregor's taxonomy [49]. Table 7 shows the TCs we found for each purpose within each area of focus. The most common purposes were analysis, explanation, and explanation & prediction, which together

Table 5. Source Papers Reporting the Development of a Theoretical Construct Based on an Existing Theory, Model, or Instrument

Theoretical construct involved in development	learning/ understanding	emotions/ attitudes/ beliefs/ self-efficacy	learning behavior/ strategies	performance/ progression/ retention	assessment/ self-assessment	errors/ misconceptions
Theory (apply, combine, extend, modify, use)	[22, 38, 61, 71, 119, 153, 182]	[7, 13, 70, 81, 124, 155, 168]	[97, 198]	[25, 180]	[4, 37, 82]	[125]
Model (revalidate, extend)			[134]			
Instrument (adapt, extend, validate)		[33, 34, 155, 167]		[10]	[4]	

Table 6. Approaches Used to Develop the Theoretical Constructs in Each of the Six Areas of Focus Based on Empirical Data and Literature

Approach used in development of TC	learning/ understanding	emotions/ attitudes/ beliefs/ self-efficacy	learning behavior/ strategies	performance/ progression/ retention	assessment/ self-assessment	errors/ misconceptions
Design-based research		[124]				
Literature, argumentation	[38, 71, 98, 127, 138, 139, 142, 149, 156, 182]	[78, 81, 103]	[24, 66]	[90, 122, 131]	[157, 173]	[114, 184]
Qualitative: classification, thematic analysis, phenomenography, grounded theory	[11, 15, 41, 139, 150, 165, 177, 179, 196]	[75, 76, 115]	[16, 18, 45, 116, 163]	[73]		[114, 125]
Quantitative: empirical, regression, path analysis, structured equation modeling, exploratory or confirmatory factor analysis, principal component analysis	[22, 61, 87, 98, 119, 127, 142, 153, 156, 182, 192]	[7, 13, 27, 28, 33, 34, 70, 78, 79, 81, 95, 96, 103, 144, 155, 167, 168, 199]	[24, 66, 97, 113, 198]	[9, 10, 21, 25, 90, 104, 122, 131, 172, 180]	[4, 37, 82, 157, 173]	[69, 114, 184]

Note that many TCs were developed using multiple techniques and approaches.

accounted for 76% of the TCs. Very few (5%) had a goal of prediction or design & action. The column N/A in the table denotes theoretically grounded instruments, which we explicitly include among our TCs, but which do not fit into Gregor's framework, because they are methodological tools.

Table 7. Purpose of the Theoretical Constructs in Each of the Six Areas of Focus

Area of Focus	Analysis	Explanation	Prediction	Explanation & prediction	Design & action	N/A
learning/understanding	10	8	2	3	0	2
emotions/attitudes/beliefs/ self-efficacy	0	4	0	11	1	6
learning behavior/strategies	2	4	0	2	1	1
performance/progression/retention	0	2	0	10	0	1
assessment/self-assessment	1	0	0	1	0	4
errors/misconceptions	3	0	0	0	0	1
Total	**16**	**18**	**2**	**27**	**2**	**15**

Note there are 80 TCs listed, as one source paper contained two TCs.

Most TCs are targeted at the university level of education (79%), with a number targeted at the school level (15%) and a few developed for post-university or workplace education (4%).

More than half the TCs were developed for single courses (55%); this was consistent across the focus areas, except for learning/understanding, where somewhat fewer than half of the TCs were developed for single courses.

5.4 [RQ4] How Have These Theoretical Constructs Been Actually Used in Further Research?

The 79 papers in our dataset were cited a total of 3,505 times according to Google Scholar on 31 December, 2020. Approximately 10% of these apparent citing papers were discarded, for the reasons given in Section 4. Almost half of the discarded papers were in languages other than English, examples being Spanish, Portuguese, Finnish, Croatian, Chinese, and Indonesian. About one-third were inaccessible to us, either because they were behind a paywall to which none of us had access, or because the Google Scholar links were dead. The remainder were discarded because they were abstracts rather than full papers or were duplicates of one form or another. The list of citations might have included slightly different links to the same paper, or it might have included links to two different versions of the same paper, such as the published one and a copy hosted by an author. One duplicate that arose frequently was a paper that had won a best paper award at a conference and was consequently reprinted in a journal.

Given the number of citing papers that remained, a complete citation analysis was beyond the scope of our resources, so, as mentioned in Section 4, if a source paper had more than about 30 citing papers, then we examined the 30 most recent, leaving us to analyze 1,727 (49%) of the citing papers. The number 30 was based, somewhat arbitrarily, on the fact that more than half the source papers (59%) had 30 or fewer citing papers.

Most of the citations are at the level of description (87%), with the remaining citing papers using the TC at the analysis (11%), construction (1%), or validation (<1%) level. Overall, only 50 (63%) of the source papers were cited at a level beyond description, leaving a substantial number of source papers (29, 36%) introducing TCs that have not been used in any further research. Seven of these were published recently (2019 or 2020), which helps to explain their lack of subsequent use. Table 8 shows the numbers of citing papers for each level of use within each area of focus.

Table 9 shows the different natures of the TCs, cross-tabulated with the different ways in which they are used in the citing papers. Analysis and explanation theories are widely used for interpreting, comparing, and explaining results (A5), as are prediction and explanation & prediction theories. The latter are also often used as frameworks to scope a study (A1). Design & action is the least used type of theory, but our dataset includes only a few TCs of this type, so it would not be wise to draw any conclusions from this finding. The column N/A denotes instruments, which

Table 8. Counts of the Levels of Use of TCs in the Citing Papers for the Six Areas of Focus

Area of Focus	Description	Analysis	Construction	Validation
learning/understanding	534	84	11	5
emotions/attitudes/beliefs/self-efficacy	457	65	5	4
performance/progression/retention	194	11	0	2
learning behavior/strategies	184	26	4	1
assessment/self-assessment	118	8	0	1
errors/misconceptions	62	5	0	0
Total	**1454**	**198**	**20**	**13**

Some citing papers cite a TC in more than one way.

Table 9. Uses of the Theoretical Constructs and Their Purposes According to Gregor's Classification [49]

Use of Construct	Analysis	Explanation	Prediction	Explanation & prediction	Design & action	N/A
A1 – framework to scope study	4	9	5	14	0	3
A2 – develop an instrument	3	4	0	1	0	15
A3 – data analysis framework	5	10	1	9	0	0
A4 – predict results	0	0	0	3	0	0
A5 – interpret/compare/explain results	15	15	11	13	3	3
A6 – design a new pedagogical method	2	5	3	4	1	0
A7 – use as an instrument	2	2	0	4	0	29
Total application	**31**	**45**	**20**	**48**	**4**	**50**
C1 – modify/extend existing construct	1	2	0	2	0	2
C2 – new construct from empirical	1	1	0	4	0	0
C3 – new construct from argumentation	4	0	1	2	0	0
Total construction	**6**	**3**	**1**	**8**	**0**	**2**
V1 – improve/discount existing theory	0	0	2	1	1	1
V2 – tested construct in new context	1	0	0	4	0	3
Total validation	**1**	**0**	**2**	**5**	**1**	**4**

are the most widely used TCs in our dataset, and which do not readily fit within Gregor's categorization of theories. The overall impression from Table 9 is that all the types of theory have been applied in some way by other researchers, and that with the possible exception of design & action theories, the different types have been used to more or less the same extent. The data reveals no clear differences in the nature of use of theories of different types.

5.5 [RQ5] What Kinds of Theory Have Been Used Very Little or Not at All?

Despite the increasing interest in developing theoretical constructs in CER, there is still limited evidence of their impact on further research. We analyzed a total of 1,727 papers citing the source papers in which the TCs had been published. Of these citing papers, only 186 (11%) used a source paper TC for analysis, construction, or validation. The rest generally cited the source paper not

Table 10. Most Widely Applied Theoretical Constructs and the Numbers of Papers in which They Are Found

Construct	Year	A1-A7	C1-C3	V1-V2	In papers	Self citations
learning edge momentum	2010	17	0	2	15	3
zone of proximal flow	2013	20	5	1	17	7
cognitive load component instrument	2014	12	2	2	16	2
computing attitudes survey	2015	20	0	0	20	1
learning trajectories, K–8	2017	15	1	0	11	5

Some citing papers use the TC in multiple ways.

because of the TC but for some other reason; for example, as related work or because of some finding of the source paper that was relevant for the citing paper.

Table 10 lists the most widely used TCs. These are the theory of the zone of proximal flow by Basawapatna et al. [7], the computing attitudes survey by Dorn and Elliott Tew [34], the learning edge momentum theory of Robins [142], K–8 learning trajectories presented by Rich et al. [139], and Morrison's cognitive load component instrument [119]. Two of these are instruments, supporting our earlier finding [108] that instruments are among the more widely used TCs.

We also checked whether the use cases might be due to authors continuing their own work. The rightmost column of Table 10 gives the number of citing papers that use the TC and that have at least one of the source paper's authors as a co-author. The low numbers of self-citations affirm that the TC has indeed been used constructively in the broader CER community.

A number of citing papers used a TC in multiple ways. For example, Hoda and Andreae [60] used the learning edge momentum theory [142] as a framework for designing new pedagogical solutions (A6) to reduce failure and attrition in CS1, but also carried out empirical evaluation to compare student results before and after applying the changes (A1, A3, A5). The reduced failure rates, among several cohorts from different programs, implied support for the theory (V1). Another example is the work by Thomas et al. [178], who analyzed students' software designs to seek support for the hypothesis that software design is a threshold skill [153] (A1, A3, A5). Their results, however, were inconclusive. Carter et al. [25] explored the use of the normalized state program model [24] and how it could be improved by integrating social aspects into the model, thus creating a more advanced social NPSM model (C2). They thus used the previous framework to support data analysis (A3), prediction of results (A4), and interpretation of the outcomes (A5). A further example is the work by Franklin et al. [46], who explored the impact of the TIPP&SEE learning strategy, which uses the zone of proximal flow [7] as one of its theoretical frameworks, for designing and evaluating their study (A1, A3, A5).

These examples demonstrate ways in which theories can support new research. However, in our pool of source papers the number of theory use cases in the citing papers was very skewed. We found only five source papers that had 10 or more citing papers building on their TC. For 60 papers, we found at most 2 citing papers building on their TCs, and this includes 31 source papers with no such citing papers at all. This encouraged us to explore what differences there might be between TCs that have been used more widely to support further research and TCs for which there is little evidence of such use. To address the obvious explanation that some of the papers were published quite recently and had not yet gathered citations, we excluded from the analysis all source papers published in 2019 and 2020, unless their TCs were used in at least 3 citing papers; this led to the inclusion of 2 source papers published in 2019. The remaining 68 papers fall into three groups: 24 papers with no use cases in their citing papers; 23 papers with 1 or 2 citing papers using their TC; and 21 papers with 3 or more citing papers using the TC.

We compared these groups to try to identify any obvious differences between them in terms of focus areas, theory type and scope, and educational level. Concerning the focus areas, the only differences worth noting were that half of the 12 TCs in the learning/understanding focus area had been used three times or more in further research, while half of the 10 papers in the performance/progression/retention focus area had not been used at all. However, as the counts in these focus areas are small, the difference might be accidental.

With regard to types of theory, we found that among 20 TCs of type explanation & prediction, only 3 had generated substantial further work, while 12 had led to no further development. A parallel observation concerns TCs that consist of a statistical model (regression or path model), based on empirical data without the use or adaptation of a supporting background theory. Only 1 of these TCs has seen substantial further development, while 7 have seen none. All of these TCs are of type explanation & prediction.

A final difference worth noting is that all 8 TCs at the K–12 level have led to some further development, with 5 of them leading to three or more such uses.

6 EXAMPLES OF THEORY DEVELOPMENT

In Sections 5.1 and 5.2, we discussed TCs that have been developed in CER and an overview of the approaches that were used in deriving them. In this section, we take a closer look at different types of approach, presenting examples of each, while explaining their nature according to Gregor's categorization. A comprehensive list would not be feasible, owing to the richness of methods used and the number of TCs; instead, we present selected examples of a number of major methodological approaches, followed by some examples of longer-term development. We hope that these examples and the corresponding source papers will guide readers to an understanding of how these methods have supported the development of TCs.

6.1 Variety of Approaches

There is a rich selection of methodologies used in developing theoretical constructs. Here, we present a number of these approaches, with examples.

6.1.1 Qualitative Approaches. The main qualitative approaches in CER have built phenomenographical outcome spaces and grounded theories. These approaches are data-driven in the sense that they analyze collected data, seeking to build an abstract description of it without basing the analysis on an existing theory or framework. Phenomenography has been used, for example, to explore students' understandings of single constructs, such as Java interfaces and software systems [15] or UML class diagrams and how they are used [16]. Some analyzed themes concern skills, such as what it means to produce a design [177], or various approaches to analysis and design [18], or the goal of learning to program [41, 165]. In addition, researchers have explored students' motives for studying [11] as well as instructors' understandings of students' success factors [73]. The common factor among phenomenographical studies is that they adopt a second-order perspective, investigating how people perceive or experience phenomena.

Phenomenographical outcome spaces are typically hierarchical categorizations that describe the variation in how the target group understands, perceives, or experiences the investigated phenomenon; they are therefore analysis-type constructs in Gregor's classification. However, the analysis is often enriched by the presentation of complementary and critical aspects of the phenomenon, which explain in more depth the differences and relationships among the categories. For example, Berglund and Eckerdal [11], when exploring students' different motives for getting a grade, describe separately the focus of the category, the dominating aspects of the students' conceptions, and how those conceptions relate to the student's responsibility for learning. Sorva et al. [163]

investigated how students experience visual program simulation as a learning method and describe separately the "act of learning"—what is it to learn—and the "why of learning." Such outcome spaces give a more comprehensive description of the categories and their relationships and thus correspond to Gregor's explanation category. There are more examples of such complex outcome spaces [73, 179].

Grounded theories also seek to describe the target of investigation at an abstract level, and their results can be an elaborated discussion of different categories and their relationships. For instance, Park et al. [125] analyze the different types of error that students make when learning HTML and CSS, categorizing them based on whether they relate to skills, rules, or knowledge, and various subcategories within these. Fitzgerald et al. [45] explore how students trace code, identifying 19 different strategies, which they further classify in several different ways to provide a versatile analysis of these strategies. These two examples are clearly analysis theories, describing "what is." More elaborated grounded theories often present a model of a system or a process. For example, Kinnunen and Simon analyze students' emotional experiences in learning to program [75, 76], resulting in a model presenting phases that students can pass through when working with exercises, encountering difficulties, and dealing with them. Rücker and Pinkwart [150] study how pre-college students understand the working of various electronic devices and the role of computing in those devices. Their grounded theory presents a constraint model for students' conceptual and reasoning processes and how they identify computing in different types of technical devices. A third example is the work of McGill [116], who analyzes the curriculum planning process for game degree programs. These three examples all provide comprehensive pictures of their targets, explaining the relevant actors and their interactions, and thus demonstrate Gregor's theories for explanation.

6.1.2 Statistical Models. In our TC pool, quantitative methods are clearly more often used than qualitative methods for developing the TCs, and often entail developing various types of statistical model, such as regression models, path models, or structural equation models.[2] Typical examples include analyzing the impact of various factors on students' success in a programming course [9, 10, 104, 172] or their attitudes [27] or frustration [144] in programming. More elaborate models, such as path models and structural equation models, explicitly model how various factors correlate with or influence one another and are typically presented as diagrams that capture the complexity of the system. For example, Wiedenbeck [192] investigates the role of previous experience, self-efficacy, and programming knowledge organization on students' performance in learning programming. Lishinski et al. [96] explore the impact of self-efficacy, meta-cognitive strategies, and goal orientation on students' performance in learning programming, building a path model of these impacts. They subsequently analyze both short-term and long-term effects of students' emotional responses to project scores, considering also the students' gender [95]. Marwan et al. [113] study how adaptive immediate feedback mediates engagement and the likelihood that students will complete a task and persist in learning computing.

Such models seek to identify the essential relationships among variables, including possible latent variables, to construct a compact presentation of their combined roles and impact on the final result; thus they fall into the category of explanation & prediction theories.

6.1.3 Data-driven Numerical Concepts. Some empirical work focuses on analyzing log data to create concepts that can present some essential features of a process in a very condensed form. For

[2]We acknowledge that it is common to use qualitative methods to develop an initial theory, followed by quantitative methods to test and validate it. However, single papers rarely present such mixed methods research as a whole. Quantitative models are often built on hypotheses combined from literature or based on work already presented elsewhere.

example, Jadud [69] explores compiler errors in students' submissions and derives an algorithm that distills their compilation behavior as a measure called error quotient, EQ. He then uses this measure as a predictor of students' success in the course. EQ thus corresponds to an instrument describing specific behavior in program development. McCall and Kölling [114] also investigate students' compilation errors, using log data collected in BlackBox [20]. They start with two obvious observations: that some errors are more common than others and that some errors are more difficult to resolve, which can be measured by examining compilation timestamps. Simply by combining these two measures, they create a new measure, error severity, which better allows errors to be ranked according to how problematic they are for students.

Both of these measures can be considered analysis-type constructs, because they describe their main target content, albeit in a very condensed form. However, they can be used in predicting results; for example, Carter et al. [25] compared the prediction power of EQ with their own work.

6.1.4 Adapting Instruments. All of the above examples begin with the data and use it as the basis for developing the theory. However, such constructs are rarely driven solely by the data. In most cases, some of the variables and factors have a theoretical background, such as self-efficacy, cognitive load, or some motivational factor, which are measured using some validated instrument.

Many such instruments developed in CER begin with an existing instrument that was developed based on a theory from psychology. Researchers in CER apply, adapt, or extend such instruments to a new context to create a new theoretical construct specific to CER. Scott and Ghinea [155] create and validate an instrument for assessing students' self-beliefs in computing, drawing on the control-value theory of achievement emotion [128]. Morrison et al. [119] adapt and validate the cognitive load component survey [85] to measure cognitive load in introductory programming. Dorn and Elliott Tew [34] adapt and validate the Colorado Learning Attitudes about Science Survey [1] to measure attitudes to studying computer science. Duran et al. [37] consider how self-evaluation can be used to create an instrument for evaluating students' performance in programming. They adapt the self-evaluation rubric of the common European framework of reference for language skills [30] into self-evaluation of programming skills and compare its performance to the SCS1 instrument [126], finding that the results are comparable. Alaoutinen [4] bases her self-evaluation instrument on Bloom's taxonomy [80] and tests its validity by comparing it with students' performance in a programming course.

Although they are grounded in theory, instruments fall beyond Gregor's categorization, as they are methodological tools.

6.1.5 Using, Adapting, and Extending Theories. While the work above focuses on adapting theory-based instruments to develop new instruments specific to the domain of computing, we identified several other approaches where existing theories were used to build a novel TC for CER. Byckling and Sajaniemi [22] developed a new analysis model to evaluate students' mental models based on their program-writing protocols. Based on the roles of variables [152], the model extends the methodology of Rist's theory of schema expansion in programming [141]. Their empirical study supports the goal that the model is generally applicable for analyzing programming knowledge at the novice level, thus falling into the analysis class of TCs.

Parkinson and Cutts [127] investigated the relationship between spatial skills and computing. After discussing various aspects of spatial skills and how they could be measured, they created a model describing how those aspects can be related to focal generation, notional machines, schemas, and the block model in the contexts of generating and comprehending programs. They investigate their hypotheses with an empirical study using several different cohorts, and the results indicate that in general, the average spatial skills of a cohort increase with academic attainment. The correlation is interesting, but the authors note that it can still be explained in multiple ways. Therefore,

we categorized this TC in the prediction category, because it can be used to predict results although the causal explanation is not clear.

The research in algorithm and program visualization tools involves theory development and extension in several phases. One of the few metastudies in computing education research was conducted by Hundhausen et al. [63], who explored factors in algorithm visualization systems that would promote students' learning results. One key finding in their work was that student engagement is an important factor for learning, and therefore that algorithm visualization systems that include some form of interaction with the learning content are more likely to improve learning results than systems in which students passively view content. This result supports the findings of a 2002 ITiCSE working group [121] that created the engagement taxonomy to classify different levels of engagement with visualization systems. The taxonomy, based on the hypothesis that more complex ways of interaction with learning content are beneficial for learning, originally described six different levels of engagement including plain viewing, responding to questions about the animation, and modifying the visualization. Myller et al. [120] subsequently presented an extended version of the taxonomy, adding more refined levels to it. In an extensive review of program visualization tools, Sorva et al. [162] used the original engagement taxonomy to classify tools. However, they found the taxonomy's dimension of mode of interaction with the system inadequate to describe the richness of features and interaction provided by the tools. In particular, the taxonomy does not assess what opportunities students have to influence the content of the visualization. Can they only work with predefined content, or give their own input or parameter values, or modify the visualized software, or even provide their own software for the tool? Combining this dimension with the original perspective of engagement led to the creation of a two-dimensional taxonomy, 2DET, which is a more accurate tool for classifying the systems (type analysis), as well as designing new tools and features in the future.

Another example of extending a theory concerns threshold concepts theory [84]. This theory postulates that disciplinary content includes certain troublesome concepts, which, when learned, irreversibly transform students' view of the discipline, permitting them to integrate their knowledge of other concepts in a new way. Computing education researchers have also explored this theory [17, 40] with the goal of identifying these threshold concepts in programming. Sanders et al. [153] extended this line of research to consider whether some computing skills could be threshold skills, thus adding a novel area for further investigation. Threshold concepts and skills fall into the category of explanation theories, because they explain the working of a complex phenomenon but cannot be used for predicting outcomes.

6.1.6 Combining Theories. When two or more theoretical perspectives are combined, it is possible to build a richer picture of the investigated target, as illustrated by the following examples:

In their theoretical paper, Kafai et al. [71] discuss different ways that computational thinking has been framed, from the perspective of skill and competence building, creative expression and participation, or social justice and ethics. They review these three framings and how they structure the theory space of computational thinking (type analysis). The combined theoretical framework, which covers cognitive, situated, and critical framings, can support computing educators to leverage the explanatory potential of the different framings for implementing and evaluating learning, teaching, and tools in computing education.

Duran et al. [38] address the challenge of evaluating how complex it is for students to comprehend program code: what aspects are involved and how they interact to increase the cognitive complexity when students are reading or tracing programs. They combine cognitive load theory [170], work in CER concerning programming schemas [141, 160, 164], and the model of hierarchical complexity, a neo-Piagetian theory concerned with the relative complexities of tasks [29]. From these

they create the **cognitive complexity of computer programs framework (CCCP)**, which allows them to generate metrics for two aspects of a program, plan depth and maximal plan interactivity, thus giving a more detailed measure for program complexity in learning (type explanation).

Next, we provide two examples of theories in the category of design & action. Nikula et al. [124] present the long-term development of an introductory programming course using a design-based research approach based on two theoretical frameworks. Considering the course as a process, they use the theory of constraints [48] from organizational science as their framework to describe the system, with the goal of identifying bottlenecks and thus increasing throughput (the pass rate). The bottlenecks themselves are considered using Herzberg's classical two-factor theory [55, 56], a theory that had been used successfully in software engineering research to identify motivational factors, which produce satisfaction in work, and hygiene factors, which produce dissatisfaction [52]. By identifying these factors in their course, and refining course practices, they implemented several iterations of the course, resulting in a clear improvement in pass rate. They summarize their principles as a new three-motivator theory: "course success can be improved by removing de-motivators, increasing intrinsic motivators, or increasing extrinsic motivators, as appropriate in the course context."

Finally, Xie et al. [194] developed a comprehensive theory of instruction for programming skills. While this TC lies in the focus area of teaching/pedagogical content knowledge in our classification, and is thus beyond the scope of our analysis, we present it here as a comprehensive example of theory development in CER. They carried out a critical discussion of a wide pool of research in programming education to establish that there are distinct programming skills: tracing, writing correct syntax, comprehending templates (reusable abstractions of programming knowledge), and writing code with templates to solve problems. However, prior theories do not translate well into concrete instruction. Based on this, they formulated a theory of instruction, based on these four programming skills across two dimensions, and identified four areas of instruction: reading semantics, writing semantics, reading templates, and writing templates. Their paper has a broad discussion of instruction, explaining how these skills are incrementally built, followed by an empirical evaluation of their approach.

6.2 Examples of Long-term Theoretical Development

We were interested in exploring examples of progressive development of theoretical constructs. Many of the constructs we found were developed through a single study, as indicated in most of the preceding examples. However, in a number of cases the TC was developed in stages, with a series of studies that progressed over time or through extending or adapting a prior TC. This was sometimes a continuation of the authors' own work and sometimes built on the work of others. We present a few examples of this type of development.

6.2.1 Progressive Development of Theory. Guzdial [50] brings together results from studies that he and his colleagues conducted on the media computation approach to teaching introductory programming. Arguing for more hypothesis-driven and theory-driven research in computing education, Guzdial reports on progress in the development of theories on the effectiveness of the media computation approach. The studies were conducted over a 10-year period, during which hypotheses were developed and tested in five theme areas: plagiarism, retention, gender, learning, and continuing computing study. This work illustrates an iterative approach to the development of theory, with new hypotheses emerging from the outcomes of studies as the theories were tested and refined. Although the success in development of theory varies across the five themes, and there is no detailed exposition of the individual theories, Guzdial suggests [50] that this work has led to deeper exploration of theory around motivation and learning.

6.2.2 Extending an Existing Theory. Work by Carter et al. [24, 25] is an example of development of a model followed by enhancement through extension of the model. Carter et al. [24] developed the **normalized programming state model (NPSM)**, which characterizes students' programming activity in terms of transitions between different states of program correctness or incorrectness (syntactic and semantic). They hypothesized that the time spent in the different states could be used to predict course performance. An empirical study found that NPSM had greater predictive power of student performance than Jadud's EQ [69] or the Watwin score [188]. Further work by the same authors extended NPSM to include student participation measures. This was motivated by social cognitive theory [6, 148], which emphasizes the value to learning of regular participation in a learning community. The resultant model, **social NPSM (SNPSM)**, was tested empirically and showed an improvement in predictive power over the original NPSM. They conclude [25] that the results affirm the importance of social interactions to success in learning.

6.2.3 Combining Theories to Develop a New Theory. Basawapatna et al. [7] present a theory-based pedagogical framework for increasing students' engagement through promoting their intrinsic motivation. The zone of proximal flow combines Vygotsky's zone of proximal development theory [185] with Csikszentmihalyi's notion of flow [31]. The **zone of proximal development (ZPD)** refers to the difference between what a student can achieve on their own and what they can achieve with guidance and encouragement from others. ZPD is a key idea from Vygotsky's social development theory [186], which states that social interaction plays a fundamental role in cognitive development. Csikszentmihalyi's notion of flow describes a state of intense concentration and feeling of effortlessness that occurs when a task has the right balance between challenge and the student's skill level and proposes that this is an optimal state for learning.

An important difference between these theories is that ZPD has a social dimension while flow is about the individual. However, Basawapatna et al. see these theories as complementary, suggesting that students can be in flow, with the task challenges matching their skill, but can at the same time be in need of outside help to meet particular task challenges. They combine the ideas from ZPD and flow to develop the zone of proximal flow, whereby students will be presented with tasks that enable them to achieve a flow state when working within their ZPD. This introduces a social element to flow. The ideas of zone of proximal flow are applied to a scalable game design project that introduces students to computer science through games. Using the framework of the zone of proximal flow, students are introduced to the games in increasing level of difficulty, scaffolded by way of extra instruction, resources, and peer learning. Basawapatna et al. describe an empirical study whose findings support the notion of the zone of proximal flow and its possible effectiveness.

7 DISCUSSION

In this section, we make some general observations, consider some implications of our findings, explain some of the limitations of our work, and briefly discuss possible future work.

7.1 General Observations

Overall, we can conclude that the interest and activity in developing domain-specific theoretical constructs has increased during the 16 years covered by our survey, with an even greater increase in activity in the most recent two years, 2019–2020. This is explained in part by the double special issue "Advancing Theory about the Novice Programmer" of the CSEd journal in 2019. However, the greatest increase has taken place in TOCE, which is interesting in view of the fact that for many years TOCE published very few papers presenting theory development, and TOCE had no special issues in 2020.

The wealth of different kinds of theoretical construct in our data supports the notion that CER is based on multiple epistemological beliefs. This is highlighted by the variety of methods used as

well as the types of theoretical construct that have been developed. We have theoretical constructs that aim at predicting phenomena, constructs that offer rich understanding of people's perceptions of phenomena and/or processes, constructs that guide instructional actions, and instruments. Theories of all these types have been used subsequently by other researchers. Instruments are the most widely used theoretical constructs in our data. In addition, theoretical constructs are often used as frameworks to scope a study or interpret its results. By contrast, TCs are less often used to create new constructs, modify existing constructs, or improve or test constructs in a new context.

We found a total of 123 new CER TCs in the years 2005–2020, with 79 of those TCs in our six target focus areas that relate to students' learning and studying. This is well in line with our earlier exploration of the years 2005–2015 [107], which found 65 TCs in total and 43 in these areas.

The area with the most active theory development is learning/understanding. However, in the past 10 years there has been increasing interest in theoretical work within the focus area of emotions/attitudes/beliefs/self-efficacy; this was the only trend that we observed. These aspects are important factors influencing students' motivation and interest in studying computing, and understanding their role in the learning/studying process gives us better insights into students' success factors. A considerable share of the interest in this area concerns the development of various theoretically based instruments to measure psychological factors among students. These include instruments to evaluate students' self-beliefs in introductory programming [155], computing attitudes [34], and self-efficacy in programming [13, 167] or algorithms [33], as well as the positive youth programming development instrument [78]. This increasing interest matches the observation that a majority of TCs in this area fall into Gregor's category of explanation & prediction.

When considering the nature of theories in other focus areas, we see that analysis and explanation theories are prevalent in the focus areas of learning/understanding and learning behavior/strategies, because in these areas it is essential to build understanding of students' conceptions of computing content, as well as their study strategies and practices. Based solely on deep knowledge of these, it is possible to develop instructional methods and pedagogies to more efficiently support students. Further work can then proceed to evaluate such interventions and derive theories to predict students' performance, progression, or retention. Obviously, most TCs in that area are of the type explanation & prediction. Among our source papers, we found very few TCs of type design & action, but this might be partly because we did not more closely investigate TCs in the focus area of teaching/pedagogical content knowledge.

As our main approach to identifying TCs is data-driven, we can only report what we found in the paper pool. However, we can also comment on possible research gaps, which might merit future consideration. First, the distribution of TCs among the 11 focus areas is by no means uniform, presumably reflecting the volume of research carried out in the focus areas. We would not expect to find many TCs addressing computing education research itself, nor content/curricula/learning goals. The first area is obviously small; moreover, as most TCs are developed by analyzing empirical data, possibly combined with using existing theories, such approaches might be less straightforward to apply in the latter focus area. However, when we consider research on teaching and pedagogical content knowledge, it is easier to foresee opportunities for rich empirical research, especially as there is some evidence that this area has not been widely addressed in CER [74].

When we consider methodologies that have been applied, we note first that design-based research has been used only once in our TC pool [124]. As computing education development work is generally a long-term process that entails evaluating a course after each offering and tuning it for the next offering, one might expect more papers using this approach. This would create a promising opportunity to systematically collect data to build and tune theoretical constructs. A possible reason for the small number is that design-based research is generally longitudinal work, and longitudinal research in CER is rarely published. There is certainly much work where researchers

build on their own previous work, but few papers, apart from instrument development, summarize long-term work, and with those that do [50, 134], the work tends not to be design-based research.

A research approach that is completely absent from our data pool is ethnography, which is fairly rarely used in CER even though it has clear potential to elicit interesting results.

7.2 Strategies for Developing Theories

Section 6 presents several different ways in which theoretical constructs have been developed in CER. Here, we summarize some strategies by which new theoretical development can be structured.

(1) Qualitative data-driven methods can help us generate an abstract presentation of the target, its structure, its components, and their relationships. Phenomenographical outcome spaces typically focus on describing the variation in the target, presenting this in hierarchical categories. The description of the differences between the categories can be enriched in several ways, and phenomenographical research often applies variation theory [92] to augment the results. Grounded theory methods provide more versatile opportunities to present the essential features or operation of a complex system, process, or set of understandings as a model with components and their relationships.
(2) Statistical models seek to present the relationships among variables and to quantify their impact on the final target outcomes, thus seeking to predict these outcomes. Statistical models often use various validated instruments to measure relevant factors.
(3) Researchers in psychology, education, and other social sciences have built numerous instruments to measure various factors in human activities. Many of these are generic in nature or specific to other domains, but they can be contextualized for computing education, for example by adapting content-related terminology to match computing concepts. There are general methodological practices that can be used to validate such new contextualized instruments.
(4) Concept inventories are questionnaires designed to investigate students' conceptions or misconceptions related to a specific target domain. An example of a novel concept inventory and how it was developed is the work by Porter et al. [132], a validated concept inventory for basic data structures.
(5) Log data analysis using various statistical methods or machine learning can help to identify different factors and their relationships in student activities. Log data often includes timestamps, facilitating the exploration of phases in activities, and other temporal changes in the data.
(6) Existing theories can be adapted or extended to manage computing education contexts, providing an existing terminology and the opportunity to reflect on the findings and consider how they might align with or differ from those in other disciplines.
(7) Existing theories can be combined to present a more holistic and versatile view of the target of investigation, providing richer ways to design studies and interpret empirical findings.

Generic methodological research literature offers recommended procedures for developing and validating instruments and concept inventories, which may be one reason behind the interest in developing them in CER. Approaches such as phenomenography, grounded theory, and statistical models, as summarized above, can lead to the development of micro-level theories. However, we are not aware of any literature concerning *how to develop theories in CER*, especially theories beyond that micro level. We therefore looked to see if we could find such strategies in other disciplines that might be applicable in the CER domain.

Theories are extensively discussed in the philosophy of science [26, 146], but such discussion is beyond the scope of this article. Rather, we sought some pragmatic theory development templates that might be applicable for CER. In the context of management science, Alvesson and Kärreman [5] present an inspiring discussion of the discovery of theory from empirical data in six phases: (1) Familiarizing oneself with the setting, in which one can begin by asking questions such as "What is going on here?" One should be open for discovery rather than constrained by some specific concept or framework. (2) Encountering/constructing breakdowns in understanding: "Fieldwork should be theoretically informed but also varied and rich enough in the sense that it allows for the existence and exploration of breakdowns." Interesting breakdowns are cases that cannot be explained by available theory, not just by the individual researcher but by other members of the research community, who should understand the case. (3) Moving from breakdown to mystery; that is, formulating some preliminary interpretations of a theoretical contribution. This includes identifying the broader relevance of an empirical finding, problems with the earlier theory or explanation and hinting at a new explanation. It is essential to critique whether the unexpected finding is bound to local context, which could be resolved with more empirical work or if it generates a wider perspective for novel explanations. (4) Engaging in more systematic work: a phase with further empirical investigation, typically supported by existing theories. (5) Solving or reformulating the mystery: one develops the idea that offers "a new interpretation of the phenomenon that inspired the mystery" and formulates it to produce something with broader relevance. (6) Developing the (re)solution, implying that the new formulation is set in clear relationship with other theories.

While this "mystery approach" gives a frame for broader investigation with generic phases, a more pragmatic approach can be found in nursing science, which also deals in depth with human targets and their interaction. There is considerable literature on developing theories in this field [65]. Below, we present a set of strategies based on those presented by Meleis [118], which were simple to adapt to CER.

- *Theory-practice-theory strategy* refers to the theory development strategy where a theorist researcher begins theorizing (deduction) from an existing (mother) theory, applies the theory in computing practice, and thereafter modifies the theory on the basis of evidence from the practice (induction). Our source papers offer several examples of this approach. For example, various self-efficacy instruments were built on self-efficacy theory, the survey questions were adapted to the computing domain, and the new instruments were validated in empirical research.
- In *practice-theory strategy*, a theorist starts the theorizing from practice observations and data and develops a theory using induction from the practice experience. Examples that fit this strategy include works that analyze log data collected from students at work [24, 69, 114].
- In *research-theory strategy*, the theorizing starts from research findings followed by inductive development of a theory from the findings. Grounded theories (Section 6.1.1) are developed using this strategy, and many statistical models (Section 6.1.2) are based on collecting and analyzing data.
- *Theory-research-theory strategy* is the process through which a theorist starts theorizing from an existing theory, conducts research using the theory as the theoretical basis, and further develops the theory using induction from the research findings. This approach fits with the work by Nikula et al. [124] (Section 6.1.6), who started with two existing theories, theory of constraints from organizational science and Herzberg's two-factor theory from management science, used these to guide their empirical work for improving practice in introductory programming, and thereafter synthesized their three-motivator theory.

- *Practice-theory-research-theory strategy* refers to the process where a theorist starts theorizing from practice, develops a theory on the basis of practice experience, conducts research on the basis of that theory, and further develops the theory on the basis of the research findings. This model fits the work by Carter et al. [25] (Section 6.2.1) when they first derived the NPSM model presenting students' actions in program development and evaluated how well the model predicted course performance, then developed a measure for social behavior and its impact on learning, then combined these approaches to build the new social NPSM model, which predicted student performance better than the original model.

These examples demonstrate how theory building can be conceptualized in a number of patterns that might help future researchers to design their work. These patterns are by no means the only ones possible. In our source paper pool, we found examples where theories were built starting with literature and pure argumentation based on previous empirical work. For example, research on learning trajectories [138, 139] began with an extensive literature survey of learning goals, which were organized and synthesized to define consensus goals. This was followed by defining pathways between the goals and using literature to support these connections until a trajectory network was constructed, presenting possible student pathways towards learning the target goal. Some the of the pathways are also based on theory, in this case, constructivism. Obviously, these trajectories can guide future empirical research as well as teaching practice.

7.3 Reflections

Our overwhelming observation is that while the source papers presenting the theoretical constructs have many citations, only a small fraction of the citing papers apply the constructs, validate them, or construct further theoretical constructs based on them. Frequently during the citation paper analysis, we found that the source paper was cited as related work only or that one of its findings was mentioned without giving any argumentation relating the TC to the finding. Indeed, in many cases the purpose of the reference is obscure, when it is included in an undifferentiated list of references with no comment on the contents of the papers.

While the above explanations will make sense to many readers, we tried to identify other reasons why only a small share of TCs attract wider attention. One of our observations is that many statistical models targeted at explanation and/or prediction have received little attention. A possible reason for this is that they tend to be very specific, depending heavily on the actual research setting and the collected data, which would make it difficult to replicate the work or apply the models to analyze a different dataset. Further research might use such TCs by including some of their variables or factors when developing new models, but without making explicit the connection between the TC and the new model. Such influence has probably remained unrecognized in our work. Moreover, some variables/factors, such as gender or self-efficacy, are so commonly used that there might be no point in reporting the source when presenting a new research design. Analysis and explanation theories seem to be used more often than prediction and explanation & prediction theories to interpret/compare/explain results and to develop an instrument or data analysis framework. There are at least two possible reasons for this phenomenon. First, we found slightly fewer prediction and explanation & prediction theories than analysis and explanation theories, which might suggest that the latter types of theory are more readily accessible by researchers. Second, analysis and explanation theories often provide a rich description and a high level of abstraction on the phenomena, which researchers might find helpful when designing data analysis frameworks or interpreting results.

Here, we encounter the challenge of replication in CER. While in the natural and medical sciences replication studies have an important role in advancing research, replication research has been undervalued in computing education, as is shown in the survey by Ahadi et al. [2]. Hao et al.

[54] carried out a systematic review of replication studies over 10 years in five key computing education research venues, the journals TOCE and CSEd and the conferences ICER, SIGCSE, and ITiCSE. They found that among 2,269 papers, only 2% were replication studies according to their criteria.

We are not able to explain this phenomenon. However, the followup consequence is that it is difficult to carry out metastudies that would synthesize a larger pool of research to identify common findings that would call for explanation and thus promote theory development. One example of such work is the metastudy of algorithm visualization by Hundhausen et al. [63], which is the base on which the engagement taxonomy hypothesis was built. The hypothesis incorporated in the taxonomy has received quite extensive interest in the visualization community, although many of the results are inconsistent. However, these inconsistencies require explanation, leading to new versions of the taxonomy [120, 162] that might induce new empirical research.

Among the constructs that have been widely used, instruments form the largest group, as shown in Table 9. Not only was using them (A7) the most common use case, but many TCs were used to develop new instruments. Many of the instruments among our TCs are used to measure various psychological factors among students [13, 33, 34, 68, 78, 119, 155, 167]. Others focus on students' conceptions of computing topics such as programming skills [4, 37], object-oriented programming [82], basic data structures [132], first-year computing course topics [184], and recursion [53]. Surprisingly many of these have not yet been used in further research, which is unfortunate given their general value. We hope that researchers will consider using these and other instruments [4, 13, 37, 53, 82, 167, 184], even though some of them have not yet been comprehensively validated.

Finally, we acknowledge that computing education has one rather specific challenge that partially hinders progress in research. The field relies heavily upon tools and technologies that develop rapidly, and therefore there is constant pressure to revise the curriculum in school and university education, a situation quite different from that in several other close disciplines such as mathematics education, physics education, and science education. While of course the educational tools and pedagogical methods develop in these fields, the main curricular content is much more stable than in computing, where both the curricular content *and* the educational tools change frequently. This is one clear reason why introductory programming has been such a focus of research for decades [101]. The past 50 years have seen many changes in the programming languages used in introductory programming. These languages include Algol, Fortran, Basic, Pascal, C, C++, C#, Scheme, Java, Haskell, Scala, and Python in university courses, and various block-based languages in schools and sometimes also in universities. While most languages have much in common, students continue to struggle with their syntax [101], a challenge that is language-specific and that cannot be overcome by teaching general principles. Longer-term changes concern the teaching of programming paradigms and how they are introduced, involving sometimes heated discussions on the use of the procedural paradigm, objects-early or objects-late approaches, functional programming, and multi-paradigm languages. A change in language always requires changes in the supporting educational tools and pedagogical practices, at least at the detailed level. Thus, the practice of computing education changes rapidly, and theories that build on the analysis of practice are frequently at risk of becoming dated. Fortunately, this is not a universal truth; for example, Soloway's classic works from the 1980s concerning programming schemas are still relevant [160, 161].

7.4 Implications for Research Training

Methodological training has had an important role in doctoral education in CER because deep knowledge and skills in some qualitative and quantitative research methods, and broader understanding of the variety of methods, are important for every researcher. This is perhaps more

important in CER than in other areas, because computing curricula, which most PhD students in CER have as their background, are often thin in methodologies for empirical research, especially research with human participants. Most PhD students have even less background in theories from the social sciences or in how theories can be used to support research designs and analysis of empirical data.

The theoretical foundations of CER are similar to those of the social sciences; based on our results, we suggest that PhD students should learn to appreciate multiple understandings and manifestations of theoretical constructs. One obvious suggestion for research training is to add explicit learning outcomes and content related to various epistemological premises of research to the doctoral training curriculum or as a theme at doctoral consortia and similar workshops. Such programs would help novice researchers to see the importance of the alignment between the researcher's epistemological premises and the formulation of research questions, methodological choices, and finally the types of empirical and theoretical contributions that might be expected to emerge from the research. We hope that the survey presented in this article, and the multiple references that it presents, provide good examples for consideration in such research training actions.

Based on our experience, research training in CER has had little focus on the selection and use of theories, and we are not aware of any training that specifically addresses the development of theories for CER. We therefore suggest that such elements be included in future research training activities, despite the complexities involved. While it is clear that such activities cannot incorporate concrete realistic exercises due to the scope of such work, critical analysis and discussion of selected research literature, focusing on how theories have been used and developed, would certainly provide doctoral students with useful insights. Furthermore, such training activities could discuss how we might enrich the analysis by integrating existing and new theories and theoretical constructs. For instance, the four forms of integration (reduction, synthesis, horizontal addition, and vertical addition) introduced by Tellings [176] might furnish useful and practical ideas about how to build upon previous theories. Finally, a review of CER-related TCs as a part of doctoral training programs could be an effective way to familiarize novice researchers with the field, especially as a standard keyword search in library databases is highly unlikely to find all relevant CER instruments, models, and theories.

7.5 Limitations

This project has encompassed an exhaustive search for TCs in three highly respected and well-established publication venues in the field of computing education research. We appreciate that not all theoretical constructs in computing education were published in these three venues. For example, Becker's repeated error density [8] was presented at ACM's ITiCSE conference, and Watson, Li, and Godwin's Watwin approach to predicting the performance of novice programming students [188] was presented at IEEE's ICALT conference. We therefore do not claim that we have examined every possible TC used in the context of CER. However, it has been established that the three venues we have considered have published the highest number of papers reporting CER-specific TCs [108]. We therefore believe that concentrating on these venues is a legitimate way to limit scope while still producing a broadly representative map covering a substantial part of theory development in CER. Hence, although the search did not encompass every paper in the field, full coverage of these three key venues for the specified period yields an extensive dataset covering a broad range of topics in CER.

We have mapped the use and development of TCs in 6 of the 11 areas of focus, as noted in Table 2. The areas selected are all directly related to student learning, studying, and progression, and form a logical subgroup of the initial 11. However, analysis of the remaining 5 areas may provide further insight.

To keep the assessment task manageable for the research team, we were not able to assess every citation for each source paper, and we had to decide how the scope would be limited. In each case, at least the 30 most recent citing papers were assessed. For many papers, this did indeed cover all citations, since the majority of source papers have been cited fewer than 30 times. However, for papers with higher numbers of citations our analysis does not cover them all.

Where human judgment is involved in classification, there is always scope for differences in interpretation. In this work, all steps in identifying and classifying the source papers were carried out independently by two researchers. The differences were discussed and resolved, which also helped to fully align the raters' judgments going forward. The citation analysis was conducted by individuals, but any unclear cases were raised for discussion with other members or with the whole group. It was sometimes challenging to interpret how a TC had been used, as we had to rely on what was reported in the paper. However, any issues in the citation analysis are unlikely to change the big picture of our findings.

7.6 Future Work

In this article, we have explored how theoretical constructs have been developed in CER and how these constructs have been used to inform further research. We have identified many different ways in which theory development has been carried out. However, we would like to establish a deeper understanding of how computing education researchers actually think and work when using and developing theories. We hope to carry out a qualitative study based on interviewing researchers to build a richer picture of how theories guide their work. We also wish to better understand how researchers seek and select the theoretical frameworks that they decide to use. Is it based on their research training, previous experience with selected theories, guidance from peer researchers, or taking examples from the literature? What is their understanding of how theories can help with their research and how they might themselves build theories? This work could also provide more insights into how theoretical work in CER might be better supported.

One of the features we investigated in this article was the main goal of the developed theory using Gregor's taxonomy. While this investigation revealed some emerging trends of the meta-characteristics of TCs in CER, it would be interesting to supplement this analysis by looking at other characteristics of the developed TCs, such as what kind of theoretical approaches the developed TCs take; for instance, to what degree TCs in CER are based on sociocultural, critical, or cognitivist theories of teaching and learning. This addition would help us to identify possible blind spots in theoretical approaches.

In addition, in this article, we explore the six focus areas related to learning, studying, and progression. In future work, we will analyze the theory development and use cases in the remaining focus areas: study choice/orientation, perceptions of computer science/computing, teaching/pedagogical content knowledge, content/curriculum/learning goals, and computing education research.

In this article, we have excluded pedagogical models and frameworks and design principles from our data pool, except for a few specific cases. In our future work, we would like to include them in the analysis to better understand how they are informed by, or how they inform, theories and models in CER. Better understanding of the role of the theory in this subfield might support not only our cumulative understanding of how pedagogical models can be developed but also our understanding of why they work.

8 CONCLUSION

We have conducted an extensive search of domain-specific theoretical constructs that have been developed and published in the past 16 years in three key venues in computing education research.

We have identified a large pool of constructs and classified them in several ways to build a big picture of the way in which theoretical development in CER has been carried out. In addition, we have looked at how the theories have been used to inform further research in publications that cite the original work.

The results have revealed a wealth of work that provides a wide coverage of computing education subfields and is slowly achieving more emphasis in the CER literature. Most of the identified constructs are quite specific and could be classified as micro-level theories. There is little evidence of continued theory development that systematically builds on previous work. Moreover, the evidence of using and extending even the available valuable resources is still very thin. The field still lacks a clear tradition of theory development, especially as regards combining and developing micro-level theories to cover broader areas.

We recommend that central actors in the field, including journals and conferences focusing on computing education, explicitly solicit papers focusing on theory development. Conferences could organize workshops to discuss the use and development of theories. Research training activities in various forms could augment their content with similar goals. We hope that the whole CER community would participate in this joint endeavor to enrich our field and thus gradually build a more solid understanding of how students learn computing in different contexts and how that learning can be supported with pedagogical practices that are designed based on the theories and supported by evidence.

APPENDICES

A THEORETICAL CONSTRUCTS, OTHER THAN INSTRUMENTS, FOUND IN THE STUDY; ORDERED CHRONOLOGICALLY WITHIN AREA OF FOCUS AND SHOWING NUMBER OF CITATIONS IN GOOGLE SCHOLAR ON 31 DECEMBER, 2020

Source paper	Theoretical construct	Citations
Area of focus: assessment/self-assessment		
Sheard et al. [157]	Classification scheme to investigate characteristics of introductory programming exam questions	41
Tahaei and Noelle [173]	Logistic regression model for detecting plagiarism in programming assessments based on patterns of resubmission	10
Area of focus: computing education research		
Simon [158]	Simon's scheme for classification of literature	5
Kinnunen et al. [74]	Scheme for classifying literature based on didactic focuses of educational research	32
Malmi et al. [106]	Scheme for classifying educational research literature covering TMFI (theories, models, frameworks, instruments) and research designs	30
Area of focus: contents/curriculum/learning goals		
Goldman et al. [47]	Lists of expert-identified central concepts for programming fundamentals, discrete math, and logic design	83
Cutts et al. [32]	**Abstraction transition (AT)** taxonomy to classify the knowledge and practices required to apprentice students into the programming community	38
McGill [116]	Factors influencing game degree program creation and curriculum planning	13
Meerbaum-Salant et al. [117]	Taxonomy for evaluating learning outcomes, a combination of the revised Bloom's and SOLO taxonomies	470

(Continued)

Continued

Source paper	Theoretical construct	Citations
Werner et al. [190]	**GCS 2.0 (game computational sophistication)**: measure of the relationship between different types of building block of computer games and game computational sophistication; new version of the GCS	1
Area of focus: emotion/beliefs/attitudes/self-efficacy		
Rodrigo and Baker [144]	Regression model to predict frustration from students' interactions with their programming environment	101
Kinnunen and Simon [75]	Partial theory explaining introductory programming students' emotional experiences with programming assignments	74
Nikula et al. [124]	Three-motivator theory for introductory programming course success	63
Kinnunen and Simon [76]	Model of four stages of introductory programming students' experiences with programming assignments	66
Basawapatna et al. [7]	Zones of proximal flow: pedagogical design framework integrating Vygotsky's zone of proximal development theory with Csikszentmihalyi's ideas about flow	71
Kothiyal et al. [79]	Model of student engagement in think-pair-share activities	110
Zhang and Dang [199]	Structural equation model of student- and instructor-related actors on intention to learn web development	4
Lishinski et al. [96]	Path model showing the influence of self-efficacy, metacognitive strategies, and intrinsic goal orientations on introductory programming performance	83
Magerko et al. [103]	Theory of change describing internal student characteristics most contributing to learning to program and their influence on intention to persist in computing	34
McCartney et al. [115]	Model of students' motivations for self-directed learning of computing	28
Stout and Blaney [168]	Two regression models: one predicting intellectual belonging to the computing community and the other predicting persistence in studying computing	14
Lishinski et al. [95]	Regression model showing that self-efficacy and frustration influence performance in introductory programming assignments	29
Clarke-Midura et al. [28]	Path models of how mother and father support affect career interest in computer science	7
Chen et al. [27]	Regression models of influence of using graphical/textual language as the first programming language on attitudes towards programming and course performance	19
Kreth et al. [81]	Model showing learning self-efficacy as the dominant factor on positive perceptions of the learning environment in an online computer science degree program	2
Jin and Divitini [70]	Path model of how affinity for technology, perceived usefulness, and enjoyment are related to intentions to continue learning	0
Area of focus: errors/misconceptions		
Jadud [69]	EQ (error quotient): measure of how much a student struggles with syntax errors while programming	239
Park et al. [125]	Taxonomy of errors made in writing HTML and CSS	22
Vahrenhold and Paul [184]	Collection of validated test items designed to detect misconceptions related to algorithms and data structures courses	18

(Continued)

Continued

Source paper	Theoretical construct	Citations
McCall and Kölling [114]	Measurement of the overall severity of different types of error—frequency x difficulty (resolution time)	12
Area of focus: learning behavior/strategies		
Fitzgerald et al. [45]	Set of strategies introductory programming students use to read and understand code	70
Box [18]	Phenomenographical outcome space with four categories describing the approaches that information system analysts/designers use to do analysis and design	10
Isohanni and Knobelsdorf [66]	Model of students' working patterns when using a program visualization tool	18
Boustedt [16]	Phenomenographical outcome space describing how computer science students understand class diagrams, **unified modeling language (UML)** symbols, and relations to **object-oriented (OO)** concepts	12
Sorva et al. [163]	Phenomenographical outcome space describing how novice programmers experience visual program simulation	31
Zarb and Hughes [198]	Communication guidelines for novice pair programmers	15
Carter et al. [24]	**Normalized program state model (NPSM)**: characterization of students' programming activities in terms of the dynamically changing syntactic and semantic correctness of their programs	79
Loksa and Ko [97]	Framework to investigate self-regulation in the context of programming	41
Marwan et al. [113]	Path model showing how adaptive immediate feedback mediates engagement and likelihood of novice programmers completing a task	2
Area of focus: learning/understanding		
Hughes et al. [61]	Good's schema refined and extended to extract information about novice programmers' comprehension of concurrent programs and their confidence in the captured knowledge	11
Wiedenbeck [192]	Path model of factors of importance in learning to program (previous programming experience, perceived self-efficacy, and knowledge organisation)	179
Eckerdal et al. [41]	Phenomenographical outcome space with five categories describing novice students' understanding of what it means to learn to program	129
Berglund and Eckerdal [11]	Three phenomenographical outcome spaces, describing advanced computer science students' motives to take a project-based distributed course in computer systems: academic achievement; project and team working capacity; social competence	28
Byckling and Sajaniemi [22]	Role plan analysis model: analysis model to evaluate students' mental models of programming concepts	21
Stamouli and Huggard [165]	Two phenomenographical outcome spaces describing introductory object-oriented programming students' understanding of learning to program and program correctness	35
Lopez et al. [98]	Hierarchical model of introductory programming skills	257
Boustedt [15]	Two phenomenographical outcome spaces describing programming students' understanding of the Java interface and software system	9

(Continued)

Continued

Source paper	Theoretical construct	Citations
Robins [142]	Learning edge momentum effect: theory to explain learning progression in introductory programming	221
Thompson and Kinshuk [179]	Phenomenographical outcome space describing practitioners' understandings of the nature of an object-oriented program	4
Sanders et al. [153]	Notion of "threshold skills"	36
Rountree et al. [149]	Extension to the definition of threshold concepts to capture strategies and mental models	15
Seiter and Foreman [156]	**Progression of early computational thinking (PECT)** model: framework for understanding and assessing computational thinking in primary school	193
Lewis [87]	Initial taxonomy for describing how students understand recursion	20
Thomas et al. [177]	Phenomenographical outcome space describing students' understanding of the phenomenon "produce a design"	13
Yuen and Robbins [196]	Model of students' quantitative and computational thinking processes during learning to program in a data-driven context	18
Rich et al. [139]	Learning trajectories describing how students can learn sequence, repetition, and conditionals	61
Rich et al. [138]	Learning trajectory for the computational thinking practice of decomposition	19
Parkinson and Cutts [127]	Theoretical model for the relationship between computer science ability and spatial skills	16
Duran et al. [38]	**Cognitive complexity of computer programs (CCCP)** framework: characterizes the complexity of a program from a cognitive perspective in terms of the hierarchical structure of plans present and their interactions	9
Kafai et al. [71]	Framework comprising cognitive, situated, and critical framings of computational thinking in K–12 computer science education for different scales of analysis	23
Rücker and Pinkwart [150]	Constraint model for students' conceptual and reasoning processes related to the identification of computing in technical devices	4
Tshukudu and Cutts [182]	**PL (programming language)** transfer model proposing three levels of knowledge to consider when programmers transfer from one programming language to another	1
Area of focus: perceptions of computer science/computing		
Schulte and Knobelsdorf [154]	Model of biographical effects of computing experiences and transitions in attitudes towards computing and computer science	152
Zander et al. [197]	Model of student identities in computer science	20
Hewner [57]	Grounded theory of students' conceptions of the field of computer science	27
Lewis et al. [88]	Grounded theory of students' assessment of their fit with computer science	60
Lewis et al. [86]	Regression model on factors predicting students' sense of belonging in computer science	11
Peters [130]	Three phenomenographical outcome spaces describing how computer science and IT engineering students experience participation and problem solving in their discipline	15
Isomöttönen et al. [67]	Three thematic analysis networks presenting students' experiences of studying computing	7

(Continued)

Continued

Source paper	Theoretical construct	Citations
Blaney [14]	Regression models of factors influencing undergraduate students' perceptions of computing leadership with the moderating role of gender	0
Mahadeo et al. [105]	Logistic regression model of factors influencing construction of computing identity	6
Rücker et al. [151]	Grounded theory of secondary school students' conceptions of critical aspects of computing technology	0
Area of focus: performance/progression/retention		
Bennedsen and Caspersen [9]	Regression model of indicators of success in introductory programming	83
Bergin and Reilly [10]	Logistic regression model for predicting programming performance (another TC from this paper is reported in Appendix B)	80
Kinnunen et al. [73]	Phenomenographical outcome space describing instructors' understanding of student success/failure	61
Lewis et al. [90]	Structural equation model of factors influencing computer science retention	42
Porter et al. [131]	**Weighted learning gain (WLG)**: measurement of student learning gains for isomorphic questions in the context of peer instruction	140
Tabanao et al. [172]	Regression model to predict novice programmers' test scores from their errors encountered, time between compilations, and EQ	87
Maguire et al. [104]	Regression model of factors predicting success in the second-year data structures and algorithms course	7
Carter et al. [25]	**SNPSM (social normalized programming state model)**: extension of NPSM model through inclusion of social participation measures	27
Tomkin et al. [180]	Logistic model for prediction of student performance	4
Nasser-Abu Alhija and Levi-Eliyahu [122]	Structural equation model of factors influencing performance in advanced computer science	1
Quille and Bergin [134]	**PreSS# (predict student success)**: machine learning model that can predict student success early in an introductory programming module (extension of PreSS)	8
Burgiel et al. [21]	Regression model of factors that predict success in college computer science	0
Area of focus: study choice/orientation		
Downes and Looker [36]	Regression model of factors influencing student plans to take computing and information technology courses	23
Lewis et al. [89]	Model of factors influencing a student's decision to major in computer science	69
Hewner and Guzdial [59]	Theory (not final) of what influences students' curricular specialization choices within a computer science major	7
Rosson et al. [148]	Path model showing influences of self-efficacy and social support on information science and technology students' career orientation	96
Guzdial et al. [51]	Model of factors influencing enrollment in undergraduate introductory computing programs for gender and racial groups	57

(Continued)

Continued

Source paper	Theoretical construct	Citations
Beyer [12]	Regression model of factors predicting likelihood of taking a future computer science course and a model of factors predicting computer science grades	208
Hewner [58]	Grounded theory of how computer science students choose courses	12
Luse et al. [99]	Structural equation model of factors influencing high-school students' choice of an IT major	19
Peteranetz et al. [129]	Regression models predicting introductory computing students' performance based on perceived instrumentality and career aspirations	6
Lakanen and Kärkkäinen [83]	Models of pathways to computer science studies	11
Wanzer et al. [187]	Theory of change tested and modified in the EarSketch context	1
Weston et al. [191]	Regression model on the factors influencing female students' persistence in computing and technology-related majors in college	10
Kemp et al. [72]	Regression models of factors influencing uptake of computer science, performance in computer science subjects, and performance in other subjects	1
Ross et al. [147]	Logistic regression models of the factors influencing undergraduate students' computer science career intentions, focusing on whether there were differential effects for Black women	1
Dou et al. [35]	Regression models on the factors influencing middle school boys' computer science career intentions (another TC from this paper is reported in Appendix B)	0
Area of focus: teaching/pedagogical content knowledge		
Ihantola et al. [64]	Taxonomy to characterize effortlessness for creating algorithm visualizations	50
Carbone et al. [23]	Two phenomenographical outcome spaces, one describing IT academics' conceptions of successful teaching (3 categories), and the other describing IT academics' conceptions of unsuccessful teaching (5 categories)	33
Tutty et al. [183]	Phenomenographical outcome spaces describing how IT academics practice their teaching	21
Sorva et al. [162]	2DET engagement taxonomy, extending the engagement taxonomy for program/algorithm visualization to incorporate content ownership dimension	365
Steghöfer et al. [166]	Conceptual model for analyzing involvement of external stakeholders in university courses	10
Xie et al. [194]	Holistic theory for teaching programming based on identification of four distinct skills that novices learn incrementally	26
Ahmad et al. [3]	Gamification framework comprising gamification constructs, game design elements, and assessment	9

B INSTRUMENTS FOUND IN THE STUDY; ORDERED CHRONOLOGICALLY WITHIN AREA OF FOCUS AND SHOWING NUMBER OF CITATIONS IN GOOGLE SCHOLAR ON 31 DECEMBER, 2020

Source paper	Instrument	Citations
Area of focus: assessment/self-assessment		
Alaoutinen [4]	Taxonomy-based scale for self-evaluation of programming knowledge	15
Kunkle and Allen [82]	Instrument to measure understanding of fundamental and object-oriented programming concepts	49
Duran et al. [37]	Instrument for self-evaluation of knowledge of programming concepts	5
Porter et al. [132]	BDSI: validated concept inventory for assessing knowledge of basic data structures concepts	15
Area of focus: emotion/beliefs/attitudes/self-efficacy		
Scott and Ghinea [155]	Instrument to assess introductory programming students' self-beliefs	33
Dorn and Elliott Tew [34]	Instrument to assess attitudes toward learning computer science	48
Danielsiek et al. [33]	Instrument to assess self-efficacy in introductory algorithms concepts and skills	16
Bhardwaj [13]	Instrument to assess self-efficacy of introductory computer science students	11
Kong and Wang [78]	PYPD: instrument to measure positive qualities (confidence, connection, contribution) of students in programming education	4
Steinhorst et al. [167]	Instrument to assess self-efficacy in introductory programming concepts and skills	1
Area of focus: learning behavior/strategies		
Israel et al. [68]	**Collaborative computing observation instrument (C-COI)** for classification of students' individual and/or collaborative behaviors during computing problem solving	16
Area of focus: learning/understanding		
Morrison et al. [119]	**Computer science cognitive load component survey (CS CLCS)**: instrument to measure cognitive load components within an introductory computer science class (adapted from Leppink's cognitive load component survey)	87
Hamouda et al. [53]	Instrument to measure students' misconceptions of recursion	18
Area of focus: performance/progression/retention		
Bergin and Reilly [10]	Instrument to measure programming self-esteem (another TC from this paper is reported in Appendix A)	80
Area of focus: study choice/orientation		
Dou et al. [35]	Adaptation of a questionnaire to determine interest in and perceptions of computer science as a career (another TC from this paper is reported in Appendix A)	0
Area of focus: teaching/pedagogical content knowledge		
Yadav and Berges [195]	Instrument to measure computer science pedagogical content knowledge	11
Zhou et al. [200]	Instrument to measure high school teachers' self-efficacy in teaching computer science	1

REFERENCES

[1] Wendy K. Adams, Katherine K. Perkins, Noah S. Podolefsky, Michael Dubson, Noah D. Finkelstein, and Carl E. Wieman. 2006. New instrument for measuring student beliefs about physics and learning physics: The Colorado Learning Attitudes about Science Survey. *Phys. Rev. Special Topics–Phys. Educ. Res.* 2, 1 (2006), 010101.

[2] Alireza Ahadi, Arto Hellas, Petri Ihantola, Ari Korhonen, and Andrew Petersen. 2016. Replication in computing education research: Researcher attitudes and experiences. In *16th Koli Calling International Conference on Computing Education Research*. 2–11.

[3] Adnan Ahmad, Furkh Zeshan, Muhammad Salman Khan, Rutab Marriam, Amjad Ali, and Alia Samreen. 2020. The impact of gamification on learning outcomes of computer science majors. *Trans. Comput. Educ.* 20, 2 (2020), 1–25.

[4] Satu Alaoutinen. 2012. Evaluating the effect of learning style and student background on self-assessment accuracy. *Comput. Sci. Educ.* 22, 2 (2012), 175–198.

[5] Mats Alvesson and Dan Kärreman. 2007. Constructing mystery: Empirical matters in theory development. *Acad. Manag. Rev.* 32, 4 (2007), 1265–1281.

[6] Albert Bandura. 1994. Self-efficacy: The exercise of control. In *Encyclopedia of Human Behavior*, V. S. Ramachandran (Ed.). Vol. 4. Academic Press, 71–81.

[7] 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 *9th International Computing Education Research Conference (ICER'13)*. 67–74.

[8] Brett A. Becker. 2016. A new metric to quantify repeated compiler errors for novice programmers. In *ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'16)*. 296–301. DOI: https://doi.org/10.1145/2899415.2899463

[9] Jens Bennedsen and Michael E. Caspersen. 2005. An investigation of potential success factors for an introductory model-driven programming course. In *1st International Computing Education Research Workshop (ICER'05)*. 155–163.

[10] Susan Bergin and Ronan Reilly. 2006. Predicting introductory programming performance: A multi-institutional multivariate study. *Comput. Sci. Educ.* 16, 4 (2006), 303–323.

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

[12] 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. *Comput. Sci. Educ.* 24, 2–3 (2014), 153–192.

[13] Jyoti Bhardwaj. 2017. In search of self-efficacy: Development of a new instrument for first year computer science students. *Comput. Sci. Educ.* 27, 2 (2017), 79–99.

[14] Jennifer M. Blaney. 2020. Gender and leadership development in undergraduate computing: A closer look at women's leadership conceptions. *Comput. Sci. Educ.* 30, 4 (2020), 469–499.

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

[16] Jonas Boustedt. 2012. Students' different understandings of class diagrams. *Comput. Sci. Educ.* 22, 1 (2012), 29–62. DOI: https://doi.org/10.1080/08993408.2012.665210

[17] Jonas Boustedt, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Mark Ratcliffe, Kate Sanders, and Carol Zander. 2007. Threshold concepts in computer science: Do they exist and are they useful? *ACM SIGCSE Bull.* 39, 1 (2007), 504–508.

[18] Ilona Box. 2009. Toward an understanding of the variation in approaches to analysis and design. *Comput. Sci. Educ.* 19, 2 (2009), 93–109. DOI: https://doi.org/10.1080/08993400902909674

[19] Eric Bredo. 2006. Philosophies of educational research. In *Handbook of Complementary Methods in Education Research*, Judith Green, Gregory Camilli, and Patricia Elmore (Eds.). Routledge, 3–31.

[20] Neil Christopher Charles Brown, Michael Kölling, Davin McCall, and Ian Utting. 2014. Blackbox: A large scale repository of novice programmers' activity. In *45th ACM Technical Symposium on Computer Science Education*. 223–228.

[21] Heidi Burgiel, Philip M. Sadler, and Gerhard Sonnert. 2020. The association of high school computer science content and pedagogy with students' success in college computer science. *Trans. Comput. Educ.* 20, 2 (Apr. 2020). DOI: https://doi.org/10.1145/3381995

[22] Pauli Byckling and Jorma Sajaniemi. 2006. A role-based analysis model for the evaluation of novices' programming knowledge development. In *2nd International Computing Education Research Workshop (ICER'06)*. 85–96. DOI: https://doi.org/10.1145/1151588.1151602

[23] Angela Carbone, Linda Mannila, and Sue Fitzgerald. 2007. Computer science and IT teachers' conceptions of successful and unsuccessful teaching: A phenomenographic study. *Comput. Sci. Educ.* 17, 4 (2007), 275–299.

[24] 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'15)*. 141–150. DOI: https://doi.org/10.1145/2787622.2787710

[25] 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. *Trans. Comput. Educ.* 17, 3 (2017), 12.

[26] H. Chang, N. Cartwright, S. Psillos, and M. Curd. 2008. Routledge, London.

[27] Chen Chen, Paulina Haduong, Karen Brennan, Gerhard Sonnert, and Philip Sadler. 2019. The effects of first programming language on college students' computing attitude and achievement: A comparison of graphical and textual languages. *Comput. Sci. Educ.* 29, 1 (2019), 23–48.

[28] Jody Clarke-Midura, Frederick J. Poole, Katarina Pantic, Chongning Sun, and Vicki Allan. 2018. How mother and father support affect youths' interest in computer science. In *14th International Computing Education Research Conference (ICER'18)*. 215–222.

[29] Michael Lamport Commons, Edward James Trudeau, Sharon Anne Stein, Francis Asbury Richards, and Sharon R. Krause. 1998. Hierarchical complexity of tasks shows the existence of developmental stages. *Devel. Rev.* 18, 3 (1998), 237–278.

[30] Council of Europe. Council for Cultural Co-operation. Education Committee. Modern Languages Division. 2001. *Common European Framework of Reference for Languages: Learning, Teaching, Assessment*. Cambridge University Press.

[31] Mihaly Csikszentmihalyi. 2000. *Beyond Boredom and Anxiety*. Jossey-Bass.

[32] 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 *8th International Conference on Computing Education Research (ICER'12)*. 63–70.

[33] Holger Danielsiek, Laura Toma, and Jan Vahrenhold. 2017. An instrument to assess self-efficacy in introductory algorithms courses. In *13th International Computing Education Research Conference (ICER'17)*. 217–225.

[34] Brian Dorn and Allison Elliott Tew. 2015. Empirical validation and application of the computing attitudes survey. *Comput. Sci. Educ.* 25, 1 (2015), 1–36.

[35] Remy Dou, Karina Bhutta, Monique Ross, Laird Kramer, and Vishodana Thamotharan. 2020. The effects of computer science stereotypes and interest on middle school boys' career intentions. *Trans. Comput. Educ.* 20, 3 (2020), 1–15.

[36] Toni Downes and Dianne Looker. 2011. Factors that influence students' plans to take computing and information technology subjects in senior secondary school. *Comput. Sci. Educ.* 21, 2 (2011), 175–199.

[37] Rodrigo Duran, Jan-Mikael Rybicki, Juha Sorva, and Arto Hellas. 2019. Exploring the value of student self-evaluation in introductory programming. In *15th International Computing Education Research Conference (ICER'19)*. 121–130.

[38] Rodrigo Duran, Juha Sorva, and Sofia Leite. 2018. Towards an analysis of program complexity from a cognitive perspective. In *14th International Computing Education Research Conference (ICER'18)*. 21–30.

[39] Emile Durkheim. 1965. *The Rules of Sociological Method*. Free Press.

[40] Anna Eckerdal, Robert McCartney, Jan Erik Moström, Mark Ratcliffe, Kate Sanders, and Carol Zander. 2006. Putting threshold concepts into context in computer science education. *ACM SIGCSE Bull.* 38, 3 (2006), 103–107.

[41] Anna Eckerdal, Michael Thuné, and Anders Berglund. 2005. What does it take to learn "programming thinking"? In *1st International Computing Education Research Workshop (ICER'05)*. 135–142. DOI: https://doi.org/10.1145/1089786.1089799

[42] Barbara J. Ericson, Kantwon Rogers, Miranda Parker, Briana Morrison, and Mark Guzdial. 2016. Identifying design principles for CS teacher Ebooks through design-based research. In *12th International Computing Education Research Conference (ICER'16)*. 191–200.

[43] Peter J. Fensham. 2004. *Defining an Identity: The Evolution of Science Education as a Field of Research*, Vol. 20. Springer Science & Business Media.

[44] Sally A. Fincher and Anthony V. Robins. 2019. *The Cambridge Handbook of Computing Education Research*. Cambridge University Press.

[45] Sue Fitzgerald, Beth Simon, and Lynda Thomas. 2005. Strategies that students use to trace code: An analysis based in grounded theory. In *1st International Computing Education Research Workshop (ICER'05)*. ACM, 69–80. DOI: https://doi.org/10.1145/1089786.1089793

[46] Diana Franklin, Jean Salac, Zachary Crenshaw, Saranya Turimella, Zipporah Klain, Marco Anaya, and Cathy Thomas. 2020. Exploring student behavior using the TIPP&SEE learning strategy. In *16th International Computing Education Research Conference (ICER'20)*. 91–101.

[47] 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. *Trans. Comput. Educ.* 10, 2 (2010), 1–29.

[48] Eliyahu M. Goldratt. 1990. *Theory of Constraints*. North River Press.
[49] Shirley Gregor. 2006. The nature of theory in information systems. *MIS Quart.* 30, 6 (2006), 611–642.
[50] Mark Guzdial. 2013. Exploring hypotheses about media computation. In *9th International Computing Education Research Conference (ICER'13)*. 19–26.
[51] 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 *8th International Computing Education Research Conference (ICER'12)*. 143–150.
[52] Tracy Hall, Nathan Baddoo, Sarah Beecham, Hugh Robinson, and Helen Sharp. 2009. A systematic review of theory use in studies investigating the motivations of software engineers. *Trans. Softw. Eng. Methodol.* 18, 3 (2009), 1–29.
[53] Sally Hamouda, Stephen H. Edwards, Hicham G. Elmongui, Jeremy V. Ernst, and Clifford A. Shaffer. 2017. A basic recursion concept inventory. *Comput. Sci. Educ.* 27, 2 (2017), 121–148.
[54] Qiang Hao, David H. Smith IV, Naitra Iriumi, Michail Tsikerdekis, and Amy J. Ko. 2019. A systematic investigation of replications in computing education research. *Trans. Comput. Educ.* 19, 4 (2019), 1–18.
[55] Frederick Herzberg. 1968. *One More Time: How Do You Motivate Employees*. Harvard Business Review, Boston, MA.
[56] Frederick Herzberg. 2017. *Motivation to Work*. Routledge.
[57] Michael Hewner. 2013. Undergraduate conceptions of the field of computer science. In *9th International Computing Education Research Conference (ICER'13)*. 107–114.
[58] Michael Hewner. 2014. How CS undergraduates make course choices. In *10th International Computing Education Research Conference*. 115–122.
[59] Michael Hewner and Mark Guzdial. 2011. How CS majors select a specialization. In *7th International Computing Education Research Workshop (ICER'11)*. 11–18.
[60] Rashina Hoda and Peter Andreae. 2014. It's not them, it's us! Why computer science fails to impress many first years. In *16th Australasian Computing Education Conference (ACE'14)*. 159–162.
[61] Connor Hughes, Jim Buckley, Chris Exton, and Des O'Carroll. 2005. Towards a framework for characterising concurrent comprehension. *Comput. Sci. Educ.* 15, 1 (2005), 7–24. DOI: https://doi.org/10.1080/08993400500056522
[62] William Huitt and John Hummel. 2003. Piaget's theory of cognitive development. *Educ. Psychol. Interact.* 3, 2 (2003), 1–5.
[63] Christopher D. Hundhausen, Sarah A. Douglas, and John T. Stasko. 2002. A meta-study of algorithm visualization effectiveness. *J. Vis. Lang. Comput.* 13, 3 (2002), 259–290.
[64] Petri Ihantola, Ville Karavirta, Ari Korhonen, and Jussi Nikander. 2005. Taxonomy of effortless creation of algorithm visualizations. In *1st International Computing Education Research Workshop (ICER'05)*. 123–133.
[65] Eun-Ok Im. 2018. Theory development strategies for middle-range theories. *Adv. Nurs. Sci.* 41, 3 (2018), 275–292.
[66] Essi Isohanni and Maria Knobelsdorf. 2010. Behind the curtain: Students' use of VIP after class. In *6th International Computing Education Research Workshop (ICER'10)*. 87–96. DOI: https://doi.org/10.1145/1839594.1839610
[67] Ville Isomöttönen, Mats Daniels, Åsa Cajander, Arnold Pears, and Roger McDermott. 2019. Searching for global employability: Can students capitalize on enabling learning environments? *Trans. Comput. Educ.* 19, 2 (2019), 1–29.
[68] Maya Israel, Quentin M. Wherfel, Saadeddine Shehab, Evan A. Ramos, Adam Metzger, and George C. Reese. 2016. Assessing collaborative computing: Development of the Collaborative-Computing Observation Instrument (C-COI). *Comput. Sci. Educ.* 26, 2–3 (2016), 208–233.
[69] Matthew C. Jadud. 2006. Methods and tools for exploring novice compilation behaviour. In *2nd International Computing Education Research Workshop (ICER'06)*. 73–84.
[70] Fufen Jin and Monica Divitini. 2020. Affinity for technology and teenagers' learning intentions. In *16th International Computing Education Research Conference (ICER'20)*. 48–55.
[71] Yasmin Kafai, Chris Proctor, and Debora Lui. 2020. From theory bias to theory dialogue: Embracing cognitive, situated, and critical framings of computational thinking in K–12 CS education. *ACM Inroads* 11, 1 (2020), 44–53.
[72] Peter E. J. Kemp, Billy Wong, and Miles G. Berry. 2019. Female performance and participation in computer science: A national picture. *Trans. Comput. Educ.* 20, 1 (2019), 1–28.
[73] Päivi Kinnunen, Robert McCartney, Laurie Murphy, and Lynda Thomas. 2007. Through the eyes of instructors: A phenomenographic investigation of student success. In *3rd International Computing Education Research Workshop (ICER'07)*. 61–72.
[74] Päivi Kinnunen, Veijo Meisalo, and Lauri Malmi. 2010. Have we missed something? Identifying missing types of research in computing education. In *6th international Workshop on Computing Education Research*. 13–22.
[75] Päivi Kinnunen and Beth Simon. 2010. Experiencing programming assignments in CS1: The emotional toll. In *6th International Computing Education Research Conference (ICER'10)*. 77–86.
[76] Päivi Kinnunen and Beth Simon. 2012. My program is OK—am I? Computing freshmen's experiences of doing programming assignments. *Comput. Sci. Educ.* 22, 1 (2012), 1–28.

[77] Siu-Cheung Kong. 2017. Development and validation of a programming self-efficacy scale for senior primary school learners. In *International Conference on Computational Thinking Education*. 97–102.
[78] Siu-Cheung Kong and Yi-Qing Wang. 2019. Positive youth development from a "3Cs" programming perspective: A multi-study investigation in the university. *Comput. Sci. Educ.* 29, 4 (2019), 335–356.
[79] Aditi Kothiyal, Rwitajit Majumdar, Sahana Murthy, and Sridhar Iyer. 2013. Effect of think-pair-share in a large CS1 class: 83% sustained engagement. In *9th International Computing Education Research Conference (ICER'13)*. ACM, 137–144.
[80] David R. Krathwohl. 2002. A revision of Bloom's taxonomy: An overview. *Theor. Pract.* 41, 4 (2002), 212–218.
[81] Quintin Kreth, Mary Eve Spirou, Sarabeth Budenstein, and Julia Melkers. 2019. How prior experience and self-efficacy shape graduate student perceptions of an online learning environment in computing. *Comput. Sci. Educ.* 29, 4 (2019), 357–381.
[82] Wanda M. Kunkle and Robert B. Allen. 2016. The impact of different teaching approaches and languages on student learning of introductory programming concepts. *Trans. Comput. Educ.* 16, 1 (2016), 1–26.
[83] Antti-Jussi Lakanen and Tommi Kärkkäinen. 2019. Identifying pathways to computer science: The long-term impact of short-term game programming outreach interventions. *Trans. Comput. Educ.* 19, 3 (2019), 1–30.
[84] Ray Land, Jan H. F. Meyer, and Jan Smith. 2008. *Threshold Concepts within the Disciplines*. BRILL.
[85] Jimmie Leppink, Fred Paas, Cees P. M. Van der Vleuten, Tamara Van Gog, and Jeroen J. G. Van Merriënboer. 2013. Development of an instrument for measuring different types of cognitive load. *Behav. Rese. Meth.* 45, 4 (2013), 1058–1072.
[86] Colleen Lewis, Paul Bruno, Jonathan Raygoza, and Julia Wang. 2019. Alignment of goals and perceptions of computing predicts students' sense of belonging in computing. In *15th International Computing Education Research Conference (ICER'19)*. 11–19.
[87] Colleen M. Lewis. 2014. Exploring variation in students' correct traces of linear recursion. In *10th International Computing Education Research Conference (ICER'14)*. 67–74. DOI: https://doi.org/10.1145/2632320.2632355
[88] Colleen M. Lewis, Ruth E. Anderson, and Ken Yasuhara. 2016. "I don't code all day": Fitting in computer science when the stereotypes don't fit. In *12th International Computing Education Research Conference (ICER'16)*. 23–32.
[89] 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 *7th international Computing Education Research Workshop (ICER'11)*. 3–10.
[90] 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 *4th International Computing Education Research Workshop (ICER'08)*. 91–100.
[91] Soohyun Nam Liao, Daniel Zingaro, Michael A. Laurenzano, William G. Griswold, and Leo Porter. 2016. Lightweight, early identification of at-risk CS1 students. In *12th International Computing Education Research Conference (ICER'16)*. 123–131.
[92] Mun Ling Lo. 2012. *Variation Theory and the Improvement of Teaching and Learning*. Göteborg: Acta Universitatis Gothoburgensis.
[93] 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'16)*. 161–169.
[94] Alex Lishinski and Aman Yadav. 2019. Motivation, attitudes, and dispositions. In *The Cambridge Handbook of Computing Education Research*, Sally A. Fincher and Anthony V. Robins (Eds.). Cambridge University Press, 801–826.
[95] Alex Lishinski, Aman Yadav, and Richard Enbody. 2017. Students' emotional reactions to programming projects in introduction to programming: Measurement approach and influence on learning outcomes. In *13th International Computing Education Research Conference (ICER'17)*. 30–38.
[96] Alex Lishinski, Aman Yadav, Jon Good, and Richard Enbody. 2016. Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance. In *12th International Computing Education Research Conference (ICER'16)*. 211–220.
[97] Dastyni Loksa and Amy J. Ko. 2016. The role of self-regulation in programming problem solving process and success. In *12th International Computing Education Research Conference (ICER'16)*. 83–91.
[98] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships between reading, tracing and writing skills in introductory programming. In *4th International Computing Education Research Workshop (ICER'08)*. 101–112.
[99] 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. *Trans. Comput. Educ.* 14, 3 (2014), 1–19.
[100] Andrew Luxton-Reilly and Paul Denny. 2010. Constructive evaluation: A pedagogy of student-contributed assessment. *Comput. Sci. Educ.* 20, 2 (2010), 145–167.

[101] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory programming: A systematic literature review. In *ITiCSE Working Group Reports (ITiCSE WGR'18)*. 55–106.
[102] Peter Machamer and Michael Silberstein. 2008. *The Blackwell Guide to the Philosophy of Science*. John Wiley & Sons.
[103] Brian Magerko, Jason Freeman, Tom McKlin, Mike Reilly, Elise Livingston, Scott McCoid, and Andrea Crews-Brown. 2016. EarSketch: A STEAM-based approach for underrepresented populations in high school computer science education. *Trans. Comput. Educ.* 16, 4 (2016), 1–25.
[104] Phil Maguire, Rebecca Maguire, and Robert Kelly. 2017. Using automatic machine assessment to teach computer programming. *Comput. Sci. Educ.* 27, 3–4 (2017), 197–214.
[105] Jonathan Mahadeo, Zahra Hazari, and Geoff Potvin. 2020. Developing a computing identity framework: Understanding computer science and information technology career choice. *Trans. Comput. Educ.* 20, 1 (2020), 1–14.
[106] Lauri Malmi, Judy Sheard, 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 *6th International Computing Education Research Workshop (ICER'10)*. 3–12.
[107] Lauri Malmi, Judy Sheard, Päivi Kinnunen, Simon, and Jane Sinclair. 2019. Computing education theories: What are they and how are they used? In *15th International Computing Education Research Conference (ICER'19)*. 187–197.
[108] Lauri Malmi, Judy Sheard, Päivi Kinnunen, Simon, and Jane Sinclair. 2020. Theories and models of emotions, attitudes, and self-efficacy in the context of programming education. In *16th International Computing Education Research Conference (ICER'20)*. 36–47.
[109] 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 *10th International Computing Education Research Conference (ICER'14)*. 27–34.
[110] Lauren Margulieux, Tuba Ayer Ketenci, and Adrienne Decker. 2019. Review of measurements used in computing education research and suggestions for increasing standardization. *Comput. Sci. Educ.* 29, 1 (2019), 49–78.
[111] Lauren E. Margulieux, Brian Dorn, and Kristin A. Searle. 2019. Learning sciences for computing education. In *The Cambridge Handbook of Computing Education Research*, Sally A. Fincher and Anthony V. Robins (Eds.). Cambridge University Press, 208–230.
[112] Lauren E. Margulieux and Briana B. Morrison. 2019. Guest editorial. *Comput. Sci. Educ.* 29, 2–3 (2019), 103–105.
[113] Samiha Marwan, Ge Gao, Susan Fisk, Thomas W. Price, and Tiffany Barnes. 2020. Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In *16th International Computing Education Research Conference (ICER'20)*. 194–203.
[114] Davin McCall and Michael Kölling. 2019. A new look at novice programmer errors. *Trans. Comput. Educ.* 19, 4 (2019), 1–30.
[115] Robert McCartney, Jonas Boustedt, Anna Eckerdal, Kate Sanders, Lynda Thomas, and Carol Zander. 2016. Why computing students learn on their own: Motivation for self-directed learning of computing. *Trans. Comput. Educ.* 16, 1 (Jan. 2016).
[116] Monica M. McGill. 2012. The curriculum planning process for undergraduate game degree programs in the United Kingdom and United States. *Trans. Comput. Educ.* 12, 2 (Apr. 2012).
[117] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. 2013. Learning computer science concepts with Scratch. *Comput. Sci. Educ.* 23, 3 (2013), 239–264.
[118] Afaf Ibrahim Meleis. 1985. Theoretical nursing: Development and progress. *Amer. J. Nurs.* 85, 12 (1985), 1350.
[119] Briana B. Morrison, Brian Dorn, and Mark Guzdial. 2014. Measuring cognitive load in introductory CS: Adaptation of an instrument. In *10th International Computing Education Research Conference (ICER'14)*. 131–138. DOI: https://doi.org/10.1145/2632320.2632348
[120] Niko Myller, Roman Bednarik, Erkki Sutinen, and Mordechai Ben-Ari. 2009. Extending the engagement taxonomy: Software visualization and collaborative learning. *Trans. Comput. Educ.* 9, 1 (2009), 1–27.
[121] Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. 2002. Exploring the role of visualization and engagement in computer science education. In *ITiCSE Working Group Reports*. 131–152.
[122] Fadia Nasser-Abu Alhija and Orna Levi-Eliyahu. 2019. Modelling achievement in advanced computer science: The role of learner characteristics and perceived learning environment. *Comput. Sci. Educ.* 29, 1 (2019), 79–102.
[123] Greg L. Nelson and Amy J. Ko. 2018. On use of theory in computing education research. In *14th International Computing Education Research Conference (ICER'18)*. 31–39.
[124] Uolevi Nikula, Orlena Gotel, and Jussi Kasurinen. 2011. A motivation guided holistic rehabilitation of the first programming course. *Trans. Comput. Educ.* 11, 4 (Nov. 2011).
[125] Thomas H. Park, Ankur Saxena, Swathi Jagannath, Susan Wiedenbeck, and Andrea Forte. 2013. Towards a taxonomy of errors in HTML and CSS. In *9th International Computing Education Research Conference (ICER'13)*. ACM, 75–82.

[126] Miranda C. Parker, Mark Guzdial, and Shelly Engleman. 2016. Replication, validation, and use of a language independent CS1 knowledge assessment. In *12th International Computing Education Research Conference (ICER'16)*. 93–101.
[127] Jack Parkinson and Quintin Cutts. 2018. Investigating the relationship between spatial skills and computer science. In *14th International Computing Education Research Conference (ICER'18)*. 106–114.
[128] Reinhard Pekrun. 2006. The control-value theory of achievement emotions: assumptions, corollaries, and implications for educational research and practice. *Educ. Psychol. Rev.* 18, 4 (2006), 315–341.
[129] Markeya S. Peteranetz, Abraham E. Flanigan, Duane F. Shell, and Leen-Kiat Soh. 2016. Perceived instrumentality and career aspirations in CS1 courses: Change and relationships with achievement. In *12th International Computing Education Research Conference (ICER'16)*. 13–21.
[130] Anne-Kathrin Peters. 2018. Students' experience of participation in a discipline—A longitudinal study of computer science and IT engineering students. *Trans. Comput. Educ.* 19, 1 (2018), 1–28.
[131] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. 2011. Peer instruction: Do students really learn from peer discussion in computing? In *7th International Computing Education Research Workshop (ICER'11)*. 45–52.
[132] Leo Porter, Daniel Zingaro, Soohyun Nam Liao, Cynthia Taylor, Kevin C. Webb, Cynthia Lee, and Michael Clancy. 2019. BDSI: A validated concept inventory for basic data structures. In *15th International Computing Education Research Conference (ICER'19)*. 111–119.
[133] Thomas W. Price, Zhongxiu Liu, Veronica Cateté, and Tiffany Barnes. 2017. Factors influencing students' help-seeking behavior while programming with human and computer tutors. In *13th International Computing Education Research Conference (ICER'17)*. 127–135.
[134] Keith Quille and Susan Bergin. 2019. CS1: How will they do? How can we help? A decade of research and practice. *Comput. Sci. Educ.* 29, 2–3 (2019), 254–282.
[135] Vennila Ramalingam and Susan Wiedenbeck. 1998. Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *J. Educ. Comput. Res.* 19, 4 (1998), 367–381.
[136] Scott Reeves, Mathieu Albert, Ayelet Kuper, and Brian David Hodges. 2008. Why use theories in qualitative research? *Brit. Med. J.* 337 (2008).
[137] Research Council of Norway. 2012. *The Role of Theory in Educational Research – Report from the March Seminar 2011.*
[138] Kathryn M. Rich, T. Andrew Binkowski, Carla Strickland, and Diana Franklin. 2018. Decomposition: A K–8 computational thinking learning trajectory. In *14th International Computing Education Research Conference (ICER'18)*. 124–132.
[139] Kathryn M. Rich, Carla Strickland, T. Andrew Binkowski, Cheryl Moran, and Diana Franklin. 2018. K–8 learning trajectories derived from research literature: Sequence, repetition, conditionals. *ACM Inroads* 9, 1 (2018), 46–55.
[140] Mark Risjord. 2014. *Philosophy of Social Science: a Contemporary Introduction*. Routledge.
[141] Robert S. Rist. 1989. Schema creation in programming. *Cog. Sci.* 13, 3 (1989), 389–414.
[142] Anthony Robins. 2010. Learning edge momentum: A new account of outcomes in CS1. *Comput. Sci. Educ.* 20, 1 (2010), 37–71.
[143] Anthony V. Robins and Lauren E. Margulieux. 2019. Cognitive sciences for computing education. In *The Cambridge Handbook of Computing Education Research*, Sally A. Fincher and Anthony V. Robins (Eds.). Cambridge University Press, 231–275.
[144] Ma Mercedes T. Rodrigo and Ryan Baker. 2009. Coarse-grained detection of student frustration in an introductory programming course. In *5th international Computing Education Research Workshop (ICER'09)*. 75–80.
[145] Alexander Rosenberg. 2008. *Philosophy of Social Science, third edition*. Westview Press Boulder.
[146] Alex Rosenberg and Lee McIntyre. 2019. *Philosophy of Science: A Contemporary Introduction*. Routledge.
[147] Monique Ross, Zahra Hazari, Gerhard Sonnert, and Philip Sadler. 2020. The intersection of being black and being a woman: Examining the effect of social computing relationships on computer science career choice. *Trans. Comput. Educ.* 20, 2 (2020), 1–15.
[148] 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. *Trans. Comput. Educ.* 11, 3 (2011).
[149] Janet Rountree, Anthony Robins, and Nathan Rountree. 2013. Elaborating on threshold concepts. *Comput. Sci. Educ.* 23, 3 (2013), 265–289. DOI: https://doi.org/10.1080/08993408.2013.834748
[150] Michael T. Rücker and Niels Pinkwart. 2018. "How else should it work?" A grounded theory of pre-college students' understanding of computing devices. *Trans. Comput. Educ.* 19, 1 (2018), 1–23.
[151] Michael T. Rücker, Wouter R. van Joolingen, and Niels Pinkwart. 2020. Small but powerful: A learning study to address secondary students' conceptions of everyday computing technology. *Trans. Comput. Educ.* 20, 2 (2020), 1–27.
[152] Jorma Sajaniemi. 2002. An empirical analysis of roles of variables in novice-level procedural programs. In *IEEE Symposia on Human Centric Computing Languages and Environments*. IEEE, 37–39.

[153] 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 *8th International Computing Education Research Conference (ICER'12)*. 23–30. DOI: https://doi.org/10.1145/2361276.2361283

[154] Carsten Schulte and Maria Knobelsdorf. 2007. Attitudes towards computer science—Computing experiences as a starting point and barrier to computer science. In *3rd International Workshop on Computing Education Research (ICER'07)*. 27–38.

[155] Michael James Scott and Gheorghita Ghinea. 2014. Measuring enrichment: The assembly and validation of an instrument to assess student self-beliefs in CS1. In *10th International Computing Education Research Conference (ICER'14)*. 123–130.

[156] Linda Seiter and Brendan Foreman. 2013. Modeling the learning progressions of computational thinking of primary grade students. In *9th International Computing Education Research Conference (ICER'13)*. 59–66. DOI: https://doi.org/10.1145/2493394.2493403

[157] 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 *7th International Computing Education Research Workshop (ICER'11)*. 33–38.

[158] Simon. 2007. A classification of recent Australasian computing education publications. *Comput. Sci. Educ.* 17, 3 (2007), 155–169.

[159] Simon. 2015. *Emergence of Computing Education as a Research Discipline*. Doctoral thesis. Aalto University. Retrieved from http://urn.fi/URN:ISBN:978-952-60-6416-1.

[160] Elliot Soloway. 1986. Learning to program = learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (1986), 850–858.

[161] Elliot Soloway and Kate Ehrlich. 1984. Empirical studies of programming knowledge. *IEEE Trans. Softw. Eng.* 5 (1984), 595–609.

[162] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *Trans. Comput. Educ.* 13, 4 (2013), 1–64.

[163] Juha Sorva, Jan Lönnberg, and Lauri Malmi. 2013. Students' ways of experiencing visual program simulation. *Comput. Sci. Educ.* 23, 3 (2013), 207–238. DOI: https://doi.org/10.1080/08993408.2013.807962

[164] James C. Spohrer, Elliot Soloway, and Edgar Pope. 1985. A goal/plan analysis of buggy Pascal programs. *Hum.–Comput. Interact.* 1, 2 (1985), 163–207.

[165] Ioanna Stamouli and Meriel Huggard. 2006. Object oriented programming and program correctness: The students' perspective. In *2nd International Computing Education Research Workshop (ICER'06)*. 109–118. DOI: https://doi.org/10.1145/1151588.1151605

[166] Jan-Philipp Steghöfer, Håkan Burden, Regina Hebig, Gul Calikli, Robert Feldt, Imed Hammouda, Jennifer Horkoff, Eric Knauss, and Grischa Liebel. 2018. Involving external stakeholders in project courses. *Trans. Comput. Educ.* 18, 2 (2018), 1–32.

[167] Phil Steinhorst, Andrew Petersen, and Jan Vahrenhold. 2020. Revisiting self-efficacy in introductory programming. In *16th International Computing Education Research Conference (ICER'20)*. 158–169.

[168] Jane G. Stout and Jennifer M. Blaney. 2017. "But it doesn't come naturally": How effort expenditure shapes the benefit of growth mindset on women's sense of intellectual belonging in computing. *Comput. Sci. Educ.* 27, 3–4 (2017), 215–228.

[169] Patrick Suppes. 1974. The place of theory in educational research. *Educ. Res.* 3, 6 (1974), 3–10.

[170] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cog. Sci.* 12, 2 (1988), 257–285.

[171] Claudia Szabo, Nickolas Falkner, Andrew Petersen, Heather Bort, Kathryn Cunningham, Peter Donaldson, Arto Hellas, James Robinson, and Judy Sheard. 2019. Review and use of learning theories within computer science education research: Primer for researchers and practitioners. In *ITiCSE Working Group Reports*. 89–109.

[172] 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 *7th International Computing Education Research Workshop (ICER'11)*. 85–92.

[173] Narjes Tahaei and David C. Noelle. 2018. Automated plagiarism detection for computer programming exercises based on patterns of resubmission. In *14th International Computing Education Research Conference (ICER'18)*. 178–186.

[174] Matti Tedre, Simon, and Lauri Malmi. 2018. Changing aims of computing education: A historical survey. *Comput. Sci. Educ.* 28, 2 (2018), 158–186.

[175] Matti Tedre and Erkki Sutinen. 2008. Three traditions of computing: What educators should know. *Comput. Sci. Educ.* 18, 3 (2008), 153–170.

[176] Agnes Tellings. 2001. Eclecticism and integration in educational theories: A metatheoretical analysis. *Educ. Theor.* 51, 3 (2001), 277.

[177] Lynda Thomas, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Kate Sanders, and Carol Zander. 2014. Graduating students' designs: Through a phenomenographic lens. In *10th International Computing Education Research Conference (ICER'14)*. 91–98. DOI: https://doi.org/10.1145/2632320.2632353

[178] 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'17)*. 34–39.

[179] Errol Thompson and Kinshuk. 2011. The nature of an object-oriented program: How do practitioners understand the nature of what they are creating? *Comput. Sci. Educ.* 21, 3 (2011), 269–287. DOI: https://doi.org/10.1080/08993408.2011.607010

[180] Jonathan H. Tomkin, Matthew West, and Geoffrey L. Herman. 2018. An improved grade point average, with applications to CS undergraduate education analytics. *Trans. Comput. Educ.* 18, 4 (2018), 1–16.

[181] Meng-Jung Tsai, Ching-Yeh Wang, and Po-Fen Hsu. 2019. Developing the computer programming self-efficacy scale for computer literacy education. *J. Educ. Comput. Res.* 56, 8 (2019), 1345–1360.

[182] Ethel Tshukudu and Quintin Cutts. 2020. Understanding conceptual transfer for students learning new programming languages. In *16th International Computing Education Research Conference (ICER'20)*. 227–237.

[183] Jodi Tutty, Judithe Sheard, and Chris Avram. 2008. Teaching in the current higher education environment: Perceptions of IT academics. *Comput. Sci. Educ.* 18, 3 (2008), 171–185.

[184] Jan Vahrenhold and Wolfgang Paul. 2014. Developing and validating test items for first-year computer science courses. *Comput. Sci. Educ.* 24, 4 (2014), 304–333.

[185] Lev Vygotsky. 1987. Zone of proximal development. In *Mind in Society: the Development of Higher Psychological Processes*. Harvard University Press, 52–91.

[186] Lev Semenovich Vygotsky. 1980. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press.

[187] Dana Linnell Wanzer, Tom McKlin, Jason Freeman, Brian Magerko, and Taneisha Lee. 2020. Promoting intentions to persist in computing: An examination of six years of the EarSketch program. *Comput. Sci. Educ.* 30, 4 (2020), 394–419.

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

[189] Max Weber. 1968. *The Methodology of the Social Sciences* (4th ed.). Free Press.

[190] Linda Werner, Jill Denner, Shannon Campe, and David M. Torres. 2020. Computational sophistication of games programmed by children: A model for its measurement. *Trans. Comput. Educ.* 20, 2 (2020), 1–23.

[191] Timothy J. Weston, Wendy M. Dubow, and Alexis Kaminsky. 2019. Predicting women's persistence in computer science- and technology-related majors from high school to college. *Trans. Comput. Educ.* 20, 1 (2019), 1–16.

[192] Susan Wiedenbeck. 2005. Factors affecting the success of non-majors in learning to program. In *1st International Computing Education Research Workshop (ICER'05)*. 13–24.

[193] Timothy Williamson. 2017. Model-building in philosophy. In *Philosophy's Future: The Problem of Philosophical Progress*, Russell Blackford and Damien Broderick (Eds.). Wiley Online Library, 159–172.

[194] Benjamin Xie, Dastyni Loksa, Greg L. Nelson, Matthew J. Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Amy J. Ko. 2019. A theory of instruction for introductory programming skills. *Comput. Sci. Educ.* 29, 2–3 (2019), 205–253.

[195] Aman Yadav and Marc Berges. 2019. Computer science pedagogical content knowledge: Characterizing teacher performance. *Trans. Comput. Educ.* 19, 3 (2019), 1–24.

[196] Timothy T. Yuen and Kay A. Robbins. 2014. A qualitative study of students' computational thinking skills in a data-driven computing class. *Trans. Comput. Educ.* 14, 4 (Dec. 2014). DOI: https://doi.org/10.1145/2676660

[197] Carol Zander, Jonas Boustedt, Robert McCartney, Jan Erik Moström, Kate Sanders, and Lynda Thomas. 2009. Student transformations: Are they computer scientists yet? In *5th International Workshop on Computing Education Research (ICER'09)*. 129–140.

[198] Mark Zarb and Janet Hughes. 2015. Breaking the communication barrier: Guidelines to aid communication within pair programming. *Comput. Sci. Educ.* 25, 2 (2015), 120–151. DOI: https://doi.org/10.1080/08993408.2015.1033125

[199] Yulei (Gavin) Zhang and Yan (Mandy) Dang. 2015. Investigating essential factors on students' perceived accomplishment and enjoyment and intention to learn in web development. *Trans. Comput. Educ.* 15, 1 (Mar. 2015).

[200] Ninger Zhou, Ha Nguyen, Christian Fischer, Debra Richardson, and Mark Warschauer. 2020. High school teachers' self-efficacy in teaching computer science. *Trans. Comput. Educ.* 20, 3 (2020), 1–18.