Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use

DASTYNI LOKSA, Towson University, USA
LAUREN MARGULIEUX, Georgia State University, USA
BRETT A. BECKER, University College Dublin, Ireland
MICHELLE CRAIG, University of Toronto, Canada
PAUL DENNY, University of Auckland, New Zealand
RAYMOND PETTIT, University of Virginia, USA
JAMES PRATHER, Abilene Christian University, USA

Metacognition and self-regulation are important skills for successful learning and have been discussed and researched extensively in the general education literature for several decades. More recently, there has been growing interest in understanding how metacognitive and self-regulatory skills contribute to student success in the context of computing education. This paper presents a thorough systematic review of metacognition and self-regulation work in the context of computer programming and an in-depth discussion of the theories that have been leveraged in some way. We also discuss several prominent metacognitive and self-regulation theories from the literature outside of computing education – for example, from psychology and education – that have yet to be applied in the context of programming education.

In our investigation, we built a comprehensive corpus of papers on metacognition and self-regulation in programming education, and then employed backward snowballing to provide a deeper examination of foundational theories from outside computing education, some of which have been explored in programming education, and others that have yet to be but hold much promise. In addition, we make new observations about the way these theories are used by the computing education community, and present recommendations on how metacognition and self-regulation can help inform programming education in the future. In particular, we discuss exemplars of studies that have used existing theories to support their design and discussion of results as well as studies that have proposed their own metacognitive theories in the context of programming education. Readers will also find the article a useful resource for helping students in programming courses develop effective strategies for metacognition and self-regulation.

CCS Concepts: • **Social and professional topics → CS1**; • **Human-centered computing → User studies**.

Additional Key Words and Phrases: cognition; CS1; metacognition; metacognitive awareness; programming; self-regulation; cognitive control

ACM Reference Format:
Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. 2022. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.* 1, 1, Article 1 (January 2022), 32 pages. https://doi.org/10.1145/3487050

Authors' addresses: Dastyni Loksa, Towson University, Towson, USA, dloksa@towson.edu; Lauren Margulieux, Georgia State University, Atlanta, USA, lmargulieux@gsu.edu; Brett A. Becker, University College Dublin, Dublin, Ireland, brett.becker@ucd.ie; Michelle Craig, University of Toronto, Toronto, Canada, mcraig@cs.toronto.edu; Paul Denny, University of Auckland, Auckland, New Zealand, paul@cs.auckland.ac.nz; Raymond Pettit, University of Virginia, Charlottesville, USA, raymond.pettit@virginia.edu; James Prather, Abilene Christian University, Abilene, TX, USA, jrp09a@acu.edu.

1946-6226/2022/1-ART1
https://doi.org/10.1145/3487050

1 INTRODUCTION

Metacognition and self-regulation are popular areas of interest in education research broadly [83], as they are linked with successful learning. Although extensively researched in other disciplines for decades [35], the application of metacognition and self-regulation in computing education contexts is newer and less understood [75], despite increased attention in recent years. The act of computer programming, in particular, is unique enough that computing education researchers should explore how prior work on metacognition and self-regulation in other areas can be applied in this context. The limited work that has taken place on metacognition and self-regulation in the context of programming has mostly aligned with results from other domains. However, there is much more work to be done to understand how existing theory can be leveraged in the context of programming education. Two broad areas to explore for example, are whether these theories need to, or can be, adapted for the context of teaching programming specifically, and whether entirely new theories of metacognition and self-regulated learning might arise from their application in this context.

Computing education researchers hoping to explore these areas of metacognition and self-regulation, however, will find decades of literature distributed across several disciplines, including psychology, education, and learning sciences. The literature is convoluted enough that recent reviews have called for more consistency in how terms are defined within and between disciplines [83]. One of the contributions of this paper is to synthesize prior work from other disciplines to provide a resource that speeds progress towards understanding the unique role of metacognition and self-regulation in computing education. As such, we have aggregated theories and explained their similarities and differences so that computing education researchers can quickly and accurately identify theories relevant to their research questions. We also present exemplars of these theories applied in computing education research in order to guide researchers new to this area on research design. In addition, we highlight promising future directions for research that will advance our knowledge.

1.1 Aims and Approach

This paper builds on recent work by Prather et al. that surveyed research presented at ACM SIGCSE conferences through 2019 on metacognition and self-regulation in programming [75]. We expand upon that work in several ways including:

- broadening and bringing up to date the breadth of work surveyed
  - including papers from ACM SIGCSE conferences (and in-cooperation venues) from 2019, 2020, and through to the present date in 2021
  - including articles from the *ACM Transactions on Computing Education* and *Computer Science Education* journals
- deepening the analysis of metacognitive and self-regulated learning theory as applied in programming education
- exploring foundational metacognitive and self-regulation theories found and used outside of the computing education community that have influenced work within that community
- identifying other such theories that have not yet, but are well poised to be applied in computing education
- describing exemplars that display good use of theory in their motivation and/or exploration of results.

The study by Prather et al. [75] made two important contributions. First, it identified and synthesized relevant theories and measurements. Many were omitted, however, because they did not appear in the computing education literature. Second, it synthesized prior work on the

topic into helpful categories for both researchers and educators. The authors classified existing papers based on how thoroughly each utilized metacognitive theory, with 123 papers classified as mentioning metacognitive or self-regulatory theory in *passing*, 53 classified as discussing the theory in a *peripheral* way, and 31 classified as involving the theories in *depth*.

In the present work, we built upon the work of Prather et al. by bringing the search up to present date and by expanding the venues and journals searched. Our method involved building a corpus of papers starting with the 31 depth papers identified in [75]. We applied the same search criteria to the *Transactions on Computing Education* and *Computer Science Education* databases, updated the search for papers from SIGCSE conference venues, and added to the corpus all papers that met the depth criteria outlined in [75]. Finally, we snowballed backwards by one level, examining the papers that are cited in our corpus. Using citation frequency as a guide, we examined the most cited papers and further expanded our corpus with those that met the depth criteria. Section 2 provides the details of these methods and discusses the practical challenges that we encountered in conducting a systematic literature search.

Once an expanded corpus was established, we investigated the use of theories of metacognition and self-regulated learning within these papers. We discovered that the corpus papers used only four main theories – Flavell's theory of metacognition [33], Bandura's Self-Regulation model [4], Zimmerman's model of Self-Regulated Learning [100], and Pintrich and de Groot's self-regulated learning theory that connects to motivation [74]. In this paper, we distinguish between theories, which posit that constructs are related, and models, which are a type of theory that specify a process connecting constructs.

While there is no related work section in the present work because the entire paper functions as one, it should be noted that there have been several recent and important literature reviews on the use of theory in computing education that mention or discuss metacognition and self-regulation. As mentioned above, Prather et al. [75] provided a literature review, sorting and grouping previous work in computing education that utilized metacognition and self-regulation. The present work contributes beyond Prather et al.'s work by going deeper into the theories, how researchers can utilize them in their own studies, and detailed examples of these theories used well in computing education. Szabo et al. provide a recent review of theories and their relationships to each other, including metacognition [90]. While their work shows that concepts like metacognition and self-efficacy are related, the present work discusses *why* they are often related in computing education literature. Szabo et al. also did not distinguish between metacognition and self-regulation theories as we do here. Recent reviews by Malmi et al. either did not mention metacognition and self-regulation [57] or barely touch on it as a related concept to theories such as self-efficacy [58]. Due to the lack of mention of our focus, applying the developed taxonomies in Malmi et al.'s work would be beyond the scope of the present work. In this work, we go beyond these mentioned above to provide a highly detailed discussion of metacognition and self-regulation theories and the most up-to-date and in-depth analysis of their use in computing education literature.

In Section 3 we describe the basic components of each of these theories. This is followed in Section 4 by a discussion of five exemplar computing education research papers that refer to these foundational theories. Next in Section 5, we discuss two computing-specific metacognition theories and two corresponding exemplars. Finally, we expand observations about the way the existing theories (both general and computer-education specific) are used in the computing education community, and provide indications and recommendations to the community on the future directions in terms of how metacognition and self-regulation can help inform programming education. In particular, this last section includes descriptions for four additional metacognition and self-regulated learning theories that were not encountered in the computing education literature of our corpus, but are commonly used in psychological and educational research. These include

Boekaert's Adaptable Learning model [13], Efklides' Metacognitive and Affective model [27], Winne and Hadwin's self-regulated learning model [96], and Hadwin et al.'s Socially Shared Regulated Learning model [38]. Each of these theories connects metacognition and self-regulation to a related construct, such as social learning or affect, making them unique from the theories already used in computing education research and, thus, valuable to expanding our knowledge. We invite computing researchers to consider using these theories in their ongoing work.

We aim to encourage connections to constructs related to cognitive control (e.g., self-efficacy, cognitive load, and social learning) by describing theories of a number of constructs not in our corpus that interact with cognitive control. These constructs range from motivational to cognitive to social, and can be measured to better understand why an intervention focused on metacognition or self-regulation may or may not affect learning. By synthesizing work on these theories, we hope that readers interested in work on cognitive control in the context of programming education will be able to design better research plans in the future.

While the primary intended contribution of this paper is to support the research community in identifying and using theories of metacognition and self-regulation appropriate to their work, readers may also find the article a useful practical resource for effective approaches to help students in programming courses develop important metacognitive skills.

1.2 Research Questions

In this work we aim to address the following research questions:

(1) What theories have been used by the computing education research community in the literature on metacognition and self-regulation in the context of programming education?
(2) What are the most seminal and influential works on these theories to inform application of these theories in future research?
(3) In what ways are computing education researchers using theories of metacognition and self-regulation to motivate their work and interpret their findings?
(4) Which theories of metacognition and self-regulation from other disciplines such as psychology and the learning sciences are not currently being used by computing education researchers and how might the theories be applied?

2 CONDUCTING THE LITERATURE SEARCH AND BUILDING THE CORPUS

The starting point for our literature search was the corpus of papers reported in a recent prior review by Prather et al. [75]. This existing corpus comprised programming-related papers that examined metacognition or self-regulation and were published in SIGCSE-sponsored venues. To be included in the corpus, papers were required to use *metacognition* or *self-regulation* as a central theoretical basis of the work, and to measure or interpret their results using these theories. This determination was made through manual inspection, and papers satisfying this requirement were said to have been categorized as "depth" (other categories being "passing" and "peripheral"). In this study we started with the 31 depth papers. We then sought to construct a more comprehensive corpus by including papers from a wider range of computing education research venues, bringing the search up to present day, and by conducting reference searching, specifically reverse snowballing, after direct database searches. We reverse snowballed both the Prather et al. "depth" papers and those resulting from our expanded and up-to-date search. Combining a database search with snowballing has been shown to drastically increase the proportion of relevant papers found, when compared with multiple database-only searches [63]. We describe our method in detail in this section, and provide a visual overview of the process and outcomes in Figure 1.

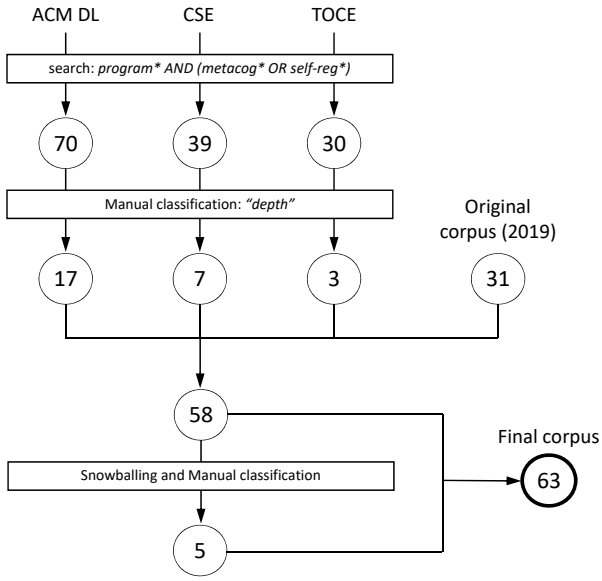

Fig. 1. A simplified schematic of the search, classification and snowball process illustrating the number of papers resulting from each step.

2.1 Search Replication (ACM DL SIGCSE-sponsored)

In their review, Prather et al. [75] conducted a literature search of SIGCSE-sponsored venues by applying the "SIGCSE-sponsored" filter to the ACM Digital Library (ACM DL) on November 26, 2019. The search string used was: "programming" AND ("metacognition" OR "self-regulation"), which was applied to the full-text content (matching anywhere in the title, abstract, keywords, article body and reference list) of all articles within the ACM Full-Text Collection.

The first step in conducting our new search was to replicate this previously documented process. However, on January 1st 2020, ACM launched an entirely new ACM DL[1], which provided slightly different search options. For instance, the inline "content.ftsec" command, which Prather et al. used to conduct the full-text search, was no longer supported. Although configuring full-text searches through the "Advanced Search" option in the new ACM DL is straightforward, we observed slightly different behavior when comparing raw results using the same search keywords after the ACM DL upgrade. We now briefly document these discrepancies, which appear to be explained by three main factors: keyword variations, in-cooperation venues, and unmatchable articles. These observations would be of interest to others who search the ACM DL and particularly those conducting systematic literature reviews using the ACM DL.

2.1.1 Keyword Variations. Previously, the ACM DL would automatically match various completions of base terms even when quoted (i.e. "metacognition"). For example, there were papers matched by the search in 2019 that contained only variations of the documented search terms, such as "metacognitive". The new ACM DL does not automatically generate these variations, however it does support wild-card matches when the search terms are not quoted. For our new search, we therefore modified the search string to the semantically equivalent: program* AND (metacog* OR self-reg*). To test the validity of this given the new search mechanism, we also manually

[1]https://libraries.acm.org/training-resources/new-dl-features

constructed a search string composed of all relevant variations of these terms: ("program" OR "programs" OR "programmers" OR "programming") AND ("metacognition" OR "metacognitive" OR "metacognitively" OR "self-regulation" OR "self-regulated" OR "self-regulatory" OR "self-regulating"). Counter-intuitively, when comparing the result sets for the wild-card and manual search strings, we found that the latter produced a *greater* number of results. Upon closer inspection, we found that two valid variations (i.e. 'self-regulatory' and 'self-regulated') of the wild-card term, self-reg*, were not being matched. Specifically, the manual search string returned papers that contained only these two variations, but the same papers were not being returned by the wild-card search. To resolve this discrepancy, the final search string included both the wild-card terms and the manually constructed terms – although most of the terms were superfluous.

2.1.2 In-cooperation Venues. On 11th March 2021 we ran our final search string on the ACM DL and applied the SIGCSE-sponsored filter, which included papers from the 2021 SIGCSE Technical Symposium. Given the search string was now (effectively) identical to the string used by Prather et al. in 2019, we expected to find all 31 of their previously categorized "depth" papers returned as a subset of the results. Surprisingly, only 24 of these 31 papers were returned. Several of the omissions appeared related to the venue of publication. In particular, ACM SIGCSE grants "in-cooperation" status to non-SIGCSE events that are sponsored by other non-profit organizations. In 2018, conferences such as Koli Calling and WiPSCE were listed as being "in-cooperation", however this designation was removed in 2019[2]. The original search, executed by Prather et al. using the SIGCSE-sponsored filter, returned several articles published in these venues prior to 2018, however our new search failed to return articles from these venues when the same filter was applied. This inconsistent behavior of the filter appears related to the changing in-cooperation status of certain venues. For consistency with the originally documented search method, we continued to apply the SIGCSE-sponsored filter when searching for newer articles.

2.1.3 Unmatchable Articles. For two of the missing seven articles that were described in Section 2.1.2, we were unable to construct a keyword query that would successfully return them. For example, one of these missing articles was published in ITiCSE 2016 (a venue "in-cooperation" with SIGCSE) and included the keywords "programming", "metacognition" and "self-regulation" multiple times. However, even searching for the paper title verbatim failed to return a matching result for this paper on the ACM DL. This issue was replicated using multiple browsers, by authors in different countries, using both incognito browser settings and when signed in to the ACM DL. Although this issue was observed over an extended period of time during the search phase, at the time of writing this article the paper in question is now successfully returned by the ACM DL search. The other missing article, a paper published at SIGCSE in 2014, still fails to be returned by our keyword search despite clearly containing the matching terms. Even other, more specific search strings, which use phrases unique to the paper, fail to return the paper in a standard search at the time of writing.

2.1.4 Results. This ACM DL search yielded a total of 70 matching articles published after November 2019. The most common venue for these articles was the SIGCSE Technical Symposium, with 25 papers in 2021 and 20 papers in 2020. The next most common was ITiCSE, with 10 papers at the conference in 2020 as well as three ITiCSE Working Group reports in 2019 and two in 2020. This was followed by ICER (9 papers in 2020), and one CompEd Working Group report in 2019.

[2]https://sigcse.org/events/incoop.html

2.2 Broadening the Search (TOCE & CSE)

To broaden the corpus beyond conference papers and working group reports, we searched the two leading computing education research journals, ACM Transactions on Computing Education (TOCE) and Computer Science Education (CSE) published by Taylor & Francis. As an ACM journal, TOCE is indexed by the ACM DL and thus we were able to use the same search string as described in Section 2.1.1. In essence, this string is: program* AND (metacog* OR self-reg*), with the manual inclusion of several term variations to avoid wild-card misses. We used the Taylor & Francis online search facility, and the same search string, for the Computer Science Education journal. Both searches were conducted on March 18th, 2021.

2.2.1 Results. Our search returned 39 unique results from Computer Science Education (46 individual papers, but including 7 pairs of duplicate papers), covering the years 1998 to 2019, and 30 results from TOCE covering the years 2005 to 2021.

2.3 Classification

These new searches resulted in a total of 139 candidate papers, which were then considered for inclusion in the corpus. Each paper was assigned to two reviewers, from the pool of seven authors, resulting in a load of either 41 or 42 papers per reviewer. Assignments were random, and balanced to minimize the number of times any two reviewers worked together. Reviewers worked independently, and considered and classified each paper. Unlike in the prior work of Prather et al., in which papers were classified into one of three categories based on the degree to which they used metacognition or self-regulation as a theoretical basis, in the current work reviewers made only a binary classification. Papers were judged on whether or not they met the requirements for the strictest category, "depth", as defined by Prather et al. [75].

All seven authors met to discuss and resolve discrepancies. For each paper where there was a disagreement over the classification, the two corresponding reviewers presented their arguments to the wider group in order to reach a consensus decision. Across all 139 papers reviewed, there were 12 disagreements (an 8% disagreement rate).

2.3.1 Results. After completing this classification phase, 17 papers from the updated ACM DL search were classified as "depth". The journal searches resulted in seven CSE and three TOCE papers at the "depth" level. These 27 new "depth" papers were added to the existing corpus of 31 papers published by Prather et al., resulting a new corpus of 58 papers.

2.4 Reverse Snowballing

The final stage in refining our corpus involved a reverse snowballing step, in which the reference list of each paper in our corpus was mined for relevant work. This stage began with manual extraction of 1789 references from the 58 corpus papers. We then calculated the frequency with which each of these referenced papers appeared across the 58 paper corpus. There were 1519 unique papers referenced in total, with only 163 being cited more than once. The most frequently cited paper appeared in the reference list of 9 articles.

We then conducted one final classification step, considering a subset of the most frequently cited papers from this snowballing process. We examined all papers that had been cited three or more times – 47 in total. We followed the same method as described in Section 2.3, randomly assigning papers to pairs of reviewers. This resulted in work from outside computing education venues and journals being added to the corpus, which are listed in Appendix A (Table 3).

2.4.1 Results. We encountered two disagreements when classifying the snowballed papers, and identified 5 new papers at the "depth" level.

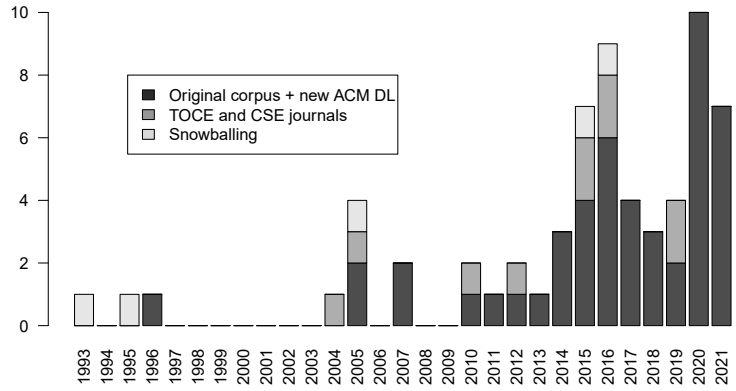

Fig. 2. Distribution of the 63 papers in the final corpus by year of publication, organized by search source.

2.5 Final Corpus

Our final corpus consisted of 63 papers, published between 1993 and 2021. The full list of papers within the corpus is detailed in Appendix A (Tables 1-3). The distribution of publication years for articles in the corpus is shown in Figure 2. We calculated the inter-rater reliability for the full two-stage rating process using Cohen's Kappa (k = 0.74, which is considered "substantial" agreement [20]).

3 THEORETICAL FRAMEWORKS USED WITHIN THE CORPUS

Metacognition, self-regulation, and self-regulated learning are terms used to describe learners' ability to plan, monitor, and evaluate their cognitive processes. They are types of cognitive control applied to make the learning process more effective or efficient [83]. Cognitive control is difficult to define and understand because it is inextricably connected to many related concepts, like self-efficacy and motivation. Furthermore, cognitive control is an internal process, which makes it difficult to measure and research. Typical measurements of internal processes, such as think aloud protocols, can be ineffective because they are inherently used during periods of high cognitive load. For these reasons, there are multiple theories of metacognition, self-regulation, and self-regulated learning that focus on different aspects of the learner or learning process, and they are all viable [67].

The distinctions between these terms are as unspecified as the definitions of each term, but there are common themes. Metacognition typically describes learners' knowledge about their cognitive control, such as which strategies are most effective for them. Self-regulation typically describes learners' process of cognitive control, such as deciding whether to take a break based on monitoring their emotion and progress. Some definitions of metacognition include self-regulation and vice versa [25], so while it is clear that these terms are separate, it is often unclear where one begins and the other ends. Self-regulated learning typically applies metacognition, self-regulation, and related constructs to a specific learning task or environment. As such, metacognition and self-regulation are typically discussed as theories (i.e., the foundations of cognitive control), and self-regulated learning is typically discussed as a framework (i.e., the application of cognitive control) [67].

To help the reader understand theories of cognitive control and the similarities and differences among them, we have created figures as visual representations of the theories being discussed. The

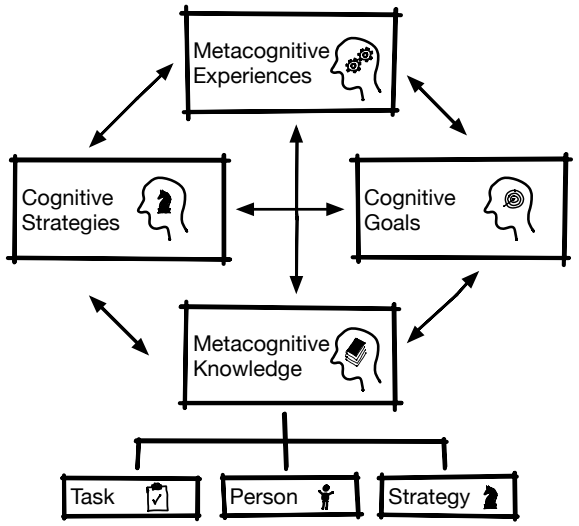

Fig. 3. Flavell Theory of Metacognition (1979)

text in the figures describes the constructs included in the theory, and we have included icons as a secondary representation of these constructs. The icons are intentionally repeated across figures that share the same constructs to make comparison easier. The text provides details of how the constructs are applied in each theory.

3.1 Foundational Theories of Cognitive Control

Theories of self-regulated learning stem from two foundational theories of cognitive control: Flavell's theory of metacognition [34] and Bandura's Social Cognitive Model of Self-Regulation [4]. Flavell was the first to use metacognition in research about cognition [33]. He did not use it explicitly in the context of education, instead describing it as "knowledge or cognitive activity that takes as its object, or regulates, any aspect of any cognitive enterprise" [34, p104]. From this definition, we see the intent to use metacognitive knowledge in the regulation of cognition, but the actual theory focuses on metacognitive knowledge's interactions with other areas of cognition and does not include a self-regulation process. This is illustrated in Figure 3.

Education researchers have applied Flavell's theory of metacognition to learning and problem-solving in education contexts [11]. It is typically described as a mechanism used by students to predict performance and monitor mastery [14]. These education researchers also explicitly linked Bandura's self-regulation model to Flavell's concept of metacognition in the context of education [11, 14].

Bandura's model of self-regulation, like Flavell's of metacognition, was also developed in general psychology, outside of the context of education. Originally, it was an addition to Bandura's Social Cognitive Theory [7] used to explain how cognitive control can overcome social and environmental predictors of behavior. This is illustrated in Figure 4. This differs from Flavell's concept of cognitive control because it defines the process of cognitive control, or self-regulation [4]. During self-regulation, a person cycles from self-observation (i.e., monitoring cognition and behavior) to judgmental processes (i.e., applying metacognitive knowledge to evaluate progress) to self-reaction (i.e., updating metacognitive knowledge about success of strategies).

Fig. 4. Bandura Social Cognitive Model of Self-Regulation (1991)

These steps are found in the many theories of self-regulated learning that have been developed after 1986 with a few key changes. Judgmental processes have been split into evaluating progress, which is grouped with monitoring, and regulating emotions and motivation, which is sometimes grouped with monitoring and sometimes treated separately [13, 27]. In addition, most self-regulated learning theories include an additional step at the beginning for planning strategy use and activating prior knowledge [73, 102]. Perhaps Bandura's model of self-regulation is a popular foundational theory for self-regulated learning because it already includes social and environmental influences on behavior, making it particularly suited to education applications.

3.2 Self-Regulated Learning Theories in Computing Education Research

The foundational theories of cognitive control, Flavell's theory of metacognition [33] and Bandura's model of self-regulation [4], are not specific to education, but they are often combined into theories of self-regulated learning. Theories of self-regulated learning often incorporate other features in addition to metacognition and self-regulation, like motivation or emotion, depending on the context. As mentioned earlier, these differences between theories have produced several co-existing theories that are used in different contexts. For this section of the paper, we will introduce two self-regulated learning theories that are commonly used in computing education, based on our review of the literature. Two additional theories that include metacognition or self-regulation, but are not yet commonly used in computing education, will be described later in the paper to complement the discussion and future work recommendations.

Both self-regulated learning theories discussed here build primarily on the process described in Bandura's model of self-regulation [4], but they also include metacognitive knowledge. These first of these theories is Zimmerman's model of self-regulated learning. This model is a three-phase process that, as illustrated in Figure 5, iterates between

- forethought (a new addition to Bandura's phases),
- performance (self-observation in Bandura's model and self-control, related to regulation of emotion and motivation in Bandura's judgmental processes phase),
- self-reflection (judgmental processes and self-reaction in Bandura's model).

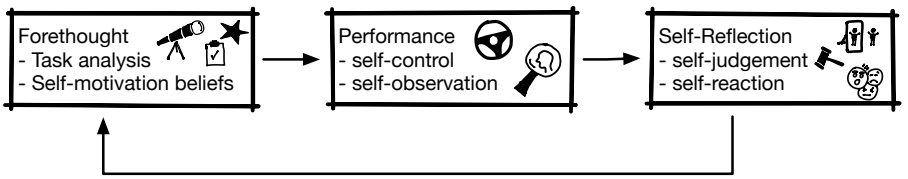

Fig. 5. Zimmerman Cyclical Phases Model (2009)

The second self-regulated learning theory commonly used in computing education differs by focusing on the relationship of motivation to self-regulation. Pintrich's [74] theory has four phases, but it also includes areas of regulation that affect all phases as illustrated in Figure 6. The four

phases are the same as Zimmerman's three, except that performance is split into monitoring or self-observation and control. Thus, the four phases are forethought, monitoring, control, and reaction and reflection. The split of performance into monitoring and control is valuable when considering the interactions between these phases and the areas of regulation. The areas of regulation are cognition, motivation and affect, behavior, and context. These areas of regulation can interact differently with monitoring than they do with control. Pintrich's theory is also unique among self-regulated learning theories in that the forethought phase explicitly includes prior knowledge activation.

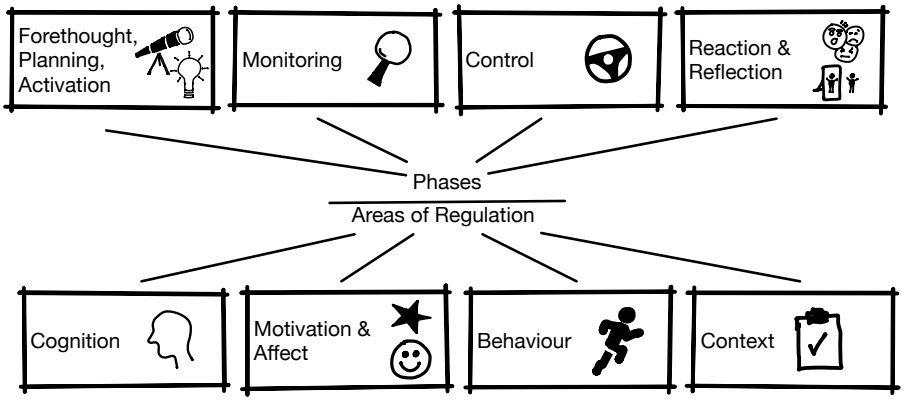

Fig. 6. Pintrich Self-Regulated Learning Theory (2000)

Computing education research draws upon additional theories related to metacognition and self-regulation that are not self-regulated learning theories. These theories include components of cognitive control, but they are not primarily about cognitive control. A common framework used in computing education is the Revised Bloom's Taxonomy [12]. While the original Bloom's taxonomy categorized learning objectives based on the depth of knowledge about the disciplinary content required, the revised taxonomy includes additional dimensions for learning objectives that are not content-dependent. One of the new dimensions in the revised taxonomy is metacognitive knowledge, which is an added knowledge dimension that learners can develop.

Another common theory seen in computing education research that includes cognitive control, and focuses on features of successful learners, is Ertmer & Newby's model of an expert learner [28]. In this model, expert learners view the learning environment as a system that includes themselves, task requirements, and strategies. Similar to the self-regulated learning phases discussed previously, this systems-view of the learners includes their metacognitive knowledge and strategies. The strategies include selecting an approach (i.e., forethought), controlling cognition and behavior (i.e., self-control), and monitoring progress towards learning goals. The only phase missing is the evaluation phase, which is arguably part of updating the metacognitive knowledge about themselves.

4 EXEMPLARS OF THEORY USE

The application of theories relating to metacognition and self-regulation appears to be of increasing interest to computing education researchers, as evidenced by publication frequency over recent years (see Figure 2). In this section, we review several recent articles that draw on knowledge from other domains and integrate metacognition and self-regulation theories into computing education

research. They serve as exemplars, demonstrating the use of theory for providing motivation for the research, using or adapting existing theoretical frameworks, and illustrating the type of valuable and transformative implications that can result.

4.1 Aligning with Zimmerman

As noted earlier, foundational theories of metacognition and self-regulation arose from outside the context of computing education. While it is helpful to understand the general self-regulated learning (SRL) strategies a student could use [101], identification of domain-specific skills is potentially of greater value. In their paper, "Identifying Computer Science Self-Regulated Learning Strategies" [32], Falkner et al. addressed this gap using a mixed-methods approach to identify SRL strategies specific to the domain of computing. They explored the research question: What SRL strategies do first-year programming students articulate as using in programming assignments and which are specific to CS?

4.1.1 Use of Theory. The authors used a grounded theory approach to understand the reflections of students in a non-introductory software engineering course. After initial coding of their reflections, the authors aligned their results with Zimmerman's SRL framework. This was done to ensure that the domain-specific SRL strategies that they identified would align well with and extend Zimmerman's existing categories. This is an excellent example of bringing external theory into computing education and customizing it for the specific characteristics of the domain.

4.1.2 Study. Using a case study approach, the authors asked 85 students to reflect twice on their software development processes and their analysis focused on how these processes changed. For their quantitative analysis, the authors reflected on the frequency of appearance of SRL strategies mentioned by students. Qualitative analysis involved the authors selecting relevant quotes from the reflections, thus framing the SRL strategies using the words of the students themselves.

4.1.3 Use of Theory in Results and Discussion. The authors presented SRL strategies in two groups along with their indicated frequencies: general SRL strategies and computing-specific SRL strategies. The areas under which the strategies were classified were: assessing difficulty, decomposing the problem, time management, personal management, and building knowledge. The skills listed under the general list were vague and could apply to most disciplines, such as "ask friends" or "time estimation" or "access resources." The strategies listed under the computing-specific list were all very specific to the domain, including "algorithm complexity" and "practice writing code." A third list provided strategies identified by students as unsuccessful, and were sorted into the same five categories. Finally, the authors reclassified all strategies using Zimmerman's categories to show into which categories computing specific SRL strategies tend to fall. Out of 841 mentions of SRL strategies by students, 243 fell into "goal-setting and planning" and 58 fell into "organizing and transforming." Curiously, none fell into two of Zimmerman's categories: "keeping records and monitoring" and "rehearsing and memorizing."

This work of Falkner et al. shows that in the domain of computing education, some SRL strategies will be more important or more relevant than others and this depends on context, such as in the early stages of learning to program where students plan before writing code and set appropriate milestones. They call for future work on scaffolding students through these early stages of programming.

4.2 Building From Zimmerman

In their paper "Evolution of Software Development Strategies" [30], Falkner et al. build on their prior work aligning their results with Zimmerman's framework. In this paper the authors investigate

which strategies students find successful and particularly relevant to their development as computer scientists and software engineers. They compare perspectives of novice and more advanced students and explore opportunities to increase reflection on, and scaffolding for, developing successful strategies.

4.2.1 Use of Theory. Like their previous paper [32] (described in Section 4.1) this work is heavily based on Zimmerman's framework (Figure 5) and explicitly identified SRL as the underlying theory. The authors define SRL and explain how the ability to "plan, set goals, organize, self-monitor and self-evaluate" are central to their efforts to identify and measure domain specific strategies for CS. Additionally, SRL strategies are central to their research questions:

(1) What CS-specific SRL strategies do final-year students articulate as using in programming assignments?
(2) What are the differences between the SRL strategies as identified by final-year students and those identified by first-year students?

The SRL strategies the authors identified are discussed extensively and, while the details are left to their prior paper, the axial coding framework that was used within this study was the one developed previously in [32].

4.2.2 Study. To investigate their research questions the authors collected data from an introductory programming course and a final year distributed systems course. In both of these courses students completed a "substantial reflective exercise that requires students to describe their current software development processes, how they have changed, and a description of how they intend to change them in the future." The authors analyzed student responses using a grounded theory approach by coding student responses and reflections, followed by axial coding using their previously developed framework. Through this analysis they identified general and CS-specific SRL strategies used by novices and by final year students, allowing for comparison between these two groups.

Their results show that CS-specific SRL strategy usage increased over the use of general SRL strategies from a ratio of 1:1 for the novices to 1.6:1 for the final year students. Successful strategies for novices tended to focus on diagram use for design, developing sub-goals, prioritization of design over implementation and testing. The final year students, in contrast, had an increased reliance on strategies such as leveraging design principles and standards, refining design, and prioritizing core components of the design over other implementation details.

4.2.3 Use of Theory in Results and Discussion. While there is no direct call back to SRL strategies in the discussion section, within their suggestions for how to scaffold some of the strategy development the authors refer to specific components of SRL, such as reflection and the sharing of mental models and strategies, as cognitive development tools. They also touch briefly on the topics of guided discovery and cognitive apprenticeship as methods that may be beneficial to support the development of SRL strategies.

4.3 Building on Bandura and Pintrich Using Pintrich's MSLQ Instrument

While some work in computing education has examined individual self-regulated learning constructs and course performance outcomes, the study by Lishinski et al. [53] examined the interactions of self-efficacy, intrinsic and extrinsic goal orientations, and metacognitive strategies and their impact on student performance in a CS1 course. In the paper titled, "Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance", the authors repeatedly take measures to capture reciprocal effects over time and develop a more complex structural model to understand how the interactions of the constructs

impact performance outcomes and the constructs themselves. This paper also looked at the effect of gender on how these constructs develop over time. The authors investigated the research questions:

(1) What are the relationships between self-efficacy, goal orientation, metacognitive strategies and course outcomes in intro programming?
(2) How does self-efficacy relate to programming performance, and how does this relationship change over time?
(3) Are there gender differences in the ways that self-efficacy affects performance in CS1?

4.3.1 Use of Theory. The authors begin the paper by situating their study in Self-Regulated Learning constructs, citing Pintrich and computing education studies that have investigated the utility of individual SRL constructs for predicting course outcomes. Beyond the high-level definition of SRL constructs, the authors provide a theoretical basis and rationale for the use of each of the constructs they are investigating. They cite Bandura [5], identifying that the strength of the self-efficacy measure often explains the otherwise weak linkages between prior ability and achievement. They provide context about the relationship of self-efficacy and other self-beliefs, identifying self-efficacy as a relevant predictor of performance due to it correlating with resiliency in difficult and novel tasks.

Citing Pintrich and Schunk, the authors define and briefly discuss the two goal orientations: intrinsic and extrinsic. They proceed to define metacognitive strategies and illustrate their importance by highlighting that appropriate cognitive strategies may still fail due to lack of metacognitive strategies. The authors describe at length the importance of, and prior work on, the relationships between these concepts which provide the basis for the model used in their study. They state that self-efficacy has been found to be positively associated with the use of more metacognitive strategies. In addition, intrinsic goal orientations have been shown to be positively associated with self-efficacy, whereas extrinsic goal orientations are negatively associated with self-efficacy. Metacognitive strategy use has also been found to be significantly related to later self-efficacy, possibly mediated by the effects of performance on self-efficacy. The authors identify prior work demonstrating that self-efficacy consistently has the strongest relationship with academic outcomes, followed by metacognitive strategies and goal orientation. Additionally they show that there exists a self-efficacy feedback loop where initial self-efficacy beliefs affect performance, which in turn affects later self-efficacy beliefs.

In addition to reviewing and providing context for each of these constructs, the authors also review prior work on these constructs specifically in computer science. They demonstrate that SRL constructs have largely confirmed theoretically expected associations within CS contexts, supporting the use of the theory outcomes in CS classes.

4.3.2 Study. The authors measured self-efficacy, metacognitive self-regulation, and intrinsic and extrinsic goal orientation using the subscales from the Motivated Strategies for Learning Questionnaire (MSLQ). The survey was given to an introductory programming course on the 2nd, 5th, and 10th week of the 12-week course. Other data collected included the scores on seven programming projects and two multiple choice exams (midterm and final).

The SRL constructs investigated have many complex interactions, and so the authors analyzed the data using a path analysis to model the relationships between the constructs. As they did with their theory usage, the authors provided explicit reasoning for their selection of analysis methods stating that path analysis is appropriate because the relationships between the motivation and learning strategies constructs are theoretically and empirically well-grounded. The results support the reciprocal relationship of the constructs with performance and are in line with the expectations

that self-efficacy would provide the strongest outcome effects and that the other MSLQ constructs would provide indirect effects.

4.3.3 Use of Theory in Results and Discussion. Perhaps because of the theory-first approach to the paper, the results are primarily discussed in terms of implications for CS pedagogy rather than theory. This is unsurprising because the results align with the expectations and in doing so provide many insights and implications for teaching. These implications include the need for a greater focus on the self-efficacy feedback loop throughout the learning process and a need to consider the gendered differences in the feedback loop effects within learning environments.

4.4 Interactions Between Metacognitive Strategies and Schemata in Program Problem-solving

In their paper, "Empirical Evidence for the Existence and Uses of Metacognition in Computer Science Problem Solving", Parham et al. utilized a think-aloud study to explore the relationship between traditional problem-solving steps (schemata) and participants' strategies for self-reflection on their progress through the problem (metacognition) [68]. The authors provided a taxonomy of schemata and metacognitive processes and then explored the interactions that occur between them while students solve a programming problem.

4.4.1 Use of Theory. The authors based their research on Flavell's theory of metacognition [34]. They were primarily interested in students' control over their own cognition and learning that they are able to display or verbalize. Flavell's theory was used as a basis for describing these interactions, which were later built upon by Sternberg [88]. The authors used these theories to study the interaction between students' metacognitive strategies, goals, and knowledge while solving a programming problem.

4.4.2 Study. After a pilot study to test validity, Parham et al. recruited 11 participants in a data structures course to solve a programming problem and observed them during the process. They followed a think-aloud protocol, observed actions taken, and interviewed students afterward. Participants' sessions were recorded on video, transcribed, and then coded for statements and actions. After reaching 85% inter-rater reliability, the researchers created their taxonomy of schemata and metacognitive processes. This taxonomy resulted in six schemata and six metacognitive processes, which they presented in a table along with definitions of each and a representative quote from a participant.

4.4.3 Use of Theory in Results and Discussion. The six schemata identified (design, write code, compile, execute, diagnose, and fix code) were all general strategies that fit neatly into classical programming problem-solving steps. The six metacognitive processes (start/revisit goals, understand the problem/plan, verbalize having low knowledge, read/consider a design, inspect the solution, and compare the solution to the problem statement) aligned with Flavell and others from psychology literature. The authors explored the relationship between the identified schemata and metacognitive processes by examining the most-common interactions between them (e.g. when a participant moves from a schemata strategy to a metacognitive one).

A common issue among novice programmers is the tendency to jump right into coding when they receive an error [8] without first re-thinking the problem or revisiting their goals [77]. Parham et al. noticed this in their study as well, and while this provided an excellent place to tie their results back to Flavell's theory, the authors missed the opportunity to do so. In terms of Flavell's theory, this would have been an example of when cognitive strategies and cognitive goals interact with actionable strategies. However, that connection does not seem to be present for many students because the researchers found that the schemata strategy of examining errors (debugging) and

the metacognitive strategy of planning/re-planning goals only interacted in 12% of recorded actions/verbalizations. This observation provides evidence that, upon seeing errors in their code, only some students will utilize a metacognitive strategy to reconsider their goals before attempting to fix the bugs in their code. Future work has continued to build on this important observation [78].

Although the authors observed that one metacognitive process could be followed by another, this was rarely the case. Instead, most visits to a metacognitive process were followed by a visit to one of the schemata, underscoring the role of metacognition in monitoring and evaluating which action to take next. Most participants, however, utilized far more schemata than metacognitive strategies. This behavior by novices is well known and underscores the need to explicitly teach metacognition to novice programmers as they learn domain knowledge [55, 76].

4.5 Complex Interactions of Self-efficacy, Motivation, and Metacognition

Many factors influence learning, and they interact in complex ways. Hull and du Boulay explored how self-efficacy, motivation, and metacognition interact to affect learner outcomes in the context of a database course [40]. Their paper, "Motivational and metacognitive feedback in SQL-Tutor*" is a good example of research that draws upon theories of metacognition and makes connections to related theoretical constructs. They modified an existing intelligent tutoring system by adding both motivational and metacognitive feedback to the user interface, and they measured the extent to which this feedback improves learner focus as well as both perceived and actual learning benefits.

4.5.1 Use of Theory. In describing the reciprocal relationship between motivation and metacognition, the authors cited theoretical work by Schunk, Pintrich and Meece in which motivation is defined as "the process whereby goal-directed activity is instigated and sustained" [84]. They further connected this to Pintrich's goal-orientation theory when describing the roles that goals have in different theories of motivation. The authors referred to the work of Paris and Winograd on metacognition in academic contexts [69], and note that according to goal-orientation theory, learners must have an insight into their own knowledge and experience in order to achieve mastery. Having provided the theoretical background for both motivation and metacognition, the authors then described how self-efficacy is deeply connected with both. They cited work by Pajares showing that self-efficacy beliefs are correlated with other motivational constructs [66], as well as work by Schunk et al. establishing that learners with high self-efficacy tend to be more likely to use various cognitive and self-regulatory learning strategies [84].

Based on the interconnections between these theories, the study asked students using an intelligent tutoring system to provide regular self-reported measures of confidence (a proxy for self-efficacy), which in turn was used to generate motivational and metacognitive feedback.

4.5.2 Study. The work reported in this paper forms part of a larger doctoral research project [39] for which the primary research question was: "What are the relationships between self-efficacy, goal orientation, metacognitive strategies and course outcomes in intro programming?". One aspect of this question was the primary focus of the paper: "Does providing such feedback lead to any measurable learning gains?". The study involved first year undergraduate students enrolled in a database course. An intelligent tutoring system, SQL-Tutor, was used as part of the course to help students develop their knowledge and skills around the construction of valid queries. Students were assigned to one of two conditions – a control condition and an intervention condition that provided additional feedback based on the past states and experiences of the learner.

The motivational and metacognitive feedback that was added to SQL-Tutor for the study was adaptive and based on learners' self-reported self-efficacy and the previous problems they had solved. At the beginning of a study session, students in the study condition were shown a short paragraph of text summarizing their previous performance and prompted to report their level of

self-efficacy. This self-report used a Likert-scale, and specifically asked students to report their "confidence" rather than self-efficacy. The authors note that these are not the same constructs, citing a discussion of the differences by Bandura [6], but argue they provided sufficient context in the question to elicit useful responses. This self-reported data was then shown to students in subsequent sessions, allowing them to reflect on how their confidence was changing. The other type of feedback shown to students in the study condition related to previously solved problems. The authors argue that reflecting on prior experience, including past problems, is important for helping students develop accurate mental representations and involves metacognitive processes such as reflection on experience in which the learner must "learn how to learn."

To address the specific research question around measurable learning gains, scores for two summative activities were compared between groups. A pre-test established that there were no significant differences between the two groups before the intervention. The post-test was the final exam. The differences between the groups as a whole were not significant. In discussing why the motivational and metacognitive feedback may not have produced measurable learning gains, as hypothesized, the authors point to several limitations in the work including a limited number of participants and a relatively short experimental time frame.

4.5.3 Use of Theory in Results and Discussion. This paper is a good example of a paper that leverages multiple related theories, in this case theories of motivation, metacognition and self-efficacy, to provide a compelling justification for the research. We found that many of the papers in our corpus used theories of metacognition and self-regulation similarly to motivate their work, but were less thorough in making connections back to the theories when interpreting or discussing their results. In discussing their results, Hull and du Boulay focus more on the practical implications of the work and potential future directions than the underlying theories. This may in part be due to the neutral results that were observed.

5 DOMAIN-SPECIFIC THEORIES RELATED TO PROGRAMMING INSTRUCTION

While most research on metacognition and self-regulation in computing education has leveraged existing theories from the fields of psychology and education, new metacognitive theories that focus on instructional approaches for teaching programming have recently emerged. Two notable examples of such theories are Xie et al.'s theory of instruction for introductory programming skills [97] and Loksa et al.'s theory of programming problem-solving [55]. The former presents reading and writing of both language semantics and reusable programming templates as four distinct skills and suggests novices may benefit by incrementally developing these skills while learning how to program. The latter proposes that novices progress through six distinct stages when engaging in problem solving while programming. Both theories provide concrete recommendations for instructional design, and they are complementary in the sense that they each target distinct skills that, together, are crucial for the successful development of programming expertise. The common metacognitive element of these theories is that making students aware of the skills or stages that they must progress through via explicit instruction empowers them to monitor their own progress and reflect on the effectiveness of the strategies and approaches they adopt when programming. Thus, both of these theories build on the earlier foundational theories outlined in Section 3 by providing direct support to aid students in monitoring their progress and evaluating their outcomes.

Xie et al. builds on the work of the BRACElet group [19], pointing to a large body of evidence that tracing, explaining, and writing code are distinct but related skills, and argues that little is known regarding how to sequence them effectively [97]. They propose a theory of instruction that sequences code tracing, writing correct syntax, reading and recognizing common code templates, and writing code using those templates. This sequence suggests reading skills are developed

before writing skills, and language semantics are mastered before common patterns of use. A novel element of the theory, compared to taxonomies and theories commonly used in computing education research, is that it provides direct and concrete recommendations for how to teach basic programming skills. The authors hypothesize that providing explicit, incremental instruction of the four skills will lower the cognitive demand on learners and result in better programming outcomes, including a more robust understanding of the relationship between individual parts of the code and its overall purpose. To test their theory, the authors conducted a small-scale randomized between-subjects experiment in which both the control and experimental conditions were provided with the same static learning materials, but for subjects in the experimental condition the four different skills were described and labeled. This explicit labeling provided subjects with a framework around which they could monitor their progress and evaluate confidence in their mastery of the related skills. Both groups engaged in practice tasks for the writing-related skills, but practice tasks for reading-related skills were provided only to subjects in the experimental condition. The total amount of practice was balanced between the groups by providing additional practice on the writing-related tasks for subjects in the control condition. In support of the theory, they found evidence that subjects in the experimental condition completed more practice exercises, made fewer errors, and exhibited a greater depth of understanding as measured by responses to metacognitive prompts appearing on the post-test.

Moving beyond the basic skills of syntax and use of existing templates, Loksa et al. present an approach for promoting metacognitive awareness in learners tackling the problem-solving aspects of programming [55]. This awareness becomes more important as tasks become more open-ended and difficult, as both novices and experts are known to exhibit more metacognitive self-regulating behaviors, such as progress monitoring, when tackling programming tasks of higher complexity. Central to Loksa et al.'s theory is the framing of programming problem-solving as an iterative process in which mental representations of problems and solutions are refined then translated and expressed as code. To help learners with this process, the authors propose explicitly teaching programming problem-solving as a sequence of six distinct stages that programmers move through, generally in order, although with some iterative refinement. These six stages directly support metacognitive awareness, as they provide learners with a concrete way to plan and monitor their progress through a problem, and to evaluate the efficacy of their strategies. When asking for help, which is a type of metacognitive control strategy, students are prompted to describe their current problem solving stage which encourages additional reflection. The authors evaluated the proposed approach using a between-subjects experiment, conducted as part of a 10-day coding camp for young children, in which an experimental group were taught the six problem solving stages and given a paper handout and physical token to aid them in tracking their progress through the stages. A control group worked on the same problems during the camp, but without receiving the explicit instruction on the problem solving stages nor being prompted to identify their stage when seeking help. Over the course of the camp, subjects in the experimental condition had higher productivity, greater independence, and improved programming self-efficacy and metacognitive awareness.

5.1 Exemplars That Use a Computing-specific Theoretical Framework

Other researchers have used Loksa's six stages as the theoretical framework for ongoing work including Prather et. al [76] who focused on the first stage, "reinterpret the prompt" in the context of an automated assessment tool. Before writing C++ code to solve a problem described in English text, 21 CS1 students in the treatment group were asked to select the expected output for a randomly-generated test case. The 17 control-group students began coding immediately after reading the problem prompt. A researcher observed all 38 students one-on-one using a think-aloud protocol as they coded their solutions. The experimental group had a higher completion rate and students

in this group solved the problem in less time and with fewer submissions than their control-group peers. However, because of the small sample size, the researchers didn't test for statistical significance of these differences. The qualitative data suggest that encouraging students to review their interpretation of the program prompt may have led to deeper metacognitive awareness. All but one student provided the correct test-case output on their first attempt which suggests that the intervention didn't necessarily correct the students' misunderstanding of the prompt, but in their verbalized comments, many students expressed an understanding of the purpose of the test case and the usefulness of the exercise. One subject explicitly commented that it "made me realize that I didn't read the problem very well because I needed to go back and read it again". The same researchers followed-up their work with a larger study and quantitative analysis to confirm their prior findings [24].

Craig et. al [22] ran a similar experiment using a different automated assessment tool on a larger set of students (n=831) but without any think-aloud interviews. In this study, CS1 students provided output to three instructor-designed test cases before writing Python code. Each student completed the test case intervention on one of two problems and acted as a control subject for the other. For one problem, the students who solved the test cases took statistically-significantly fewer attempts to submit correct Python solutions. For the other problem, the students in both groups performed the same. Based on inspection of the problem prompts and the errors submitted by the students, the authors concluded that giving students test cases can help when the students are struggling to understand the problem prompt (Loksa stage 1), but not when the students correctly understand the task and struggle with the implementation (Loksa stages 4-5).

6 DISCUSSION

6.1 Use of Theories for Motivation, Measurements, and Discussion

The papers in our corpus were selected for their use of theories of metacognition and/or self-regulation and followed the criteria outlined in [75] for being categorized as "depth" papers. We found that most papers identified theories of metacognition and/or self-regulation as a motivation for their work and cited the primary theoretical frameworks listed in Section 3. However, there were 16 papers which identified metacognition or self-regulation as the motivation of the work and either did not refer to a specific framework, or cited no cognitive theories at all. While the authors of many of these papers, specifically the ones who wrote about metacognition and self-regulation in general, offered appropriate prior work and demonstrated an understanding of the concepts of cognitive control, it is unclear what theoretical frameworks they were drawing from.

Some studies conducted measurements of participants' metacognition or self-regulation behaviors. These measurements came from a wide range of instruments including the analysis of transcriptions of participants thinking aloud, surveys, quizzes, or self-reports and reflections such as participant journals. The most predominant instrument, used by 17 papers, was the Motivated Strategies for Learning Questionnaire (MSLQ) which includes up to 12 items on the Metacognitive Self-Regulation scale.

While some of the papers measured metacognition and self-regulation, most were focused on measuring student outcomes after a given intervention and did not measure self-regulation or metacognition, despite stating the theories as motivation for the work. One reason for this may be that the authors were not familiar with the cognitive mechanisms or with how to measure them. This is a plausible reason given that interest in metacognitive theories is relatively new in computing education. Another reason why authors may not have explicitly measured self-regulation or metacognition within their studies might be that the studies may have been experimental in nature and not well suited to measuring the specifics of theoretical effects.

Another trend we identified was that papers often did not use theories when interpreting the results or did not contextualize their observations or further discussion within the framework (even if they motivated their work by referencing a self-regulation theory.) Instead, papers appropriately focused on the implications of their work and where their results can inform future work. However, not considering the results in the context of the theories or not considering the theories given the results, leaves the reader on their own to consider how the results and the theories interact. This lack of theory-based discussion may be because the authors believe that the theories, and their expectations, apply directly to the context of CS and that there exist no domain-specific nuances. Alternatively, because the papers are often focused on an intervention to improve or understand student performance in CS and not focused on how well the theories apply, it is perhaps unsurprising that there is not more discussion of theory in papers where authors must contend with page length constraints of conference venues. Whatever the reason, given the nascent nature of domain-specific use of these theories, the careful consideration of how the theories from other domains apply within a CS context is a missed opportunity to better understand our domain.

6.2 Promising Theories Underutilized in the Literature

The papers that we analyzed in this review used a subset of the common theories of self-regulated learning used in education research. This section describes theories that were rarely or not seen in this review. We hope to make researchers aware of other options with which they might not be familiar.

The first is Boekaerts' Adaptable Learning Model [13], illustrated in Figure 7. Boekaerts' model separates the self-regulation of learning and the self-regulation of motivation as competing processes. So while other theories include motivation, Boekaerts' model is unique in that it treats self-regulation of motivation as an alternative goal to learning. In other ways, the model follows a similar process as other theories. It starts with analyzing the task, which is similar to forethought, followed by monitoring and self-reaction and self-reflection. From this point, the learner can decide to regulate their cognition towards the goal of learning or to regulate their motivation towards the goal of maintaining well-being.

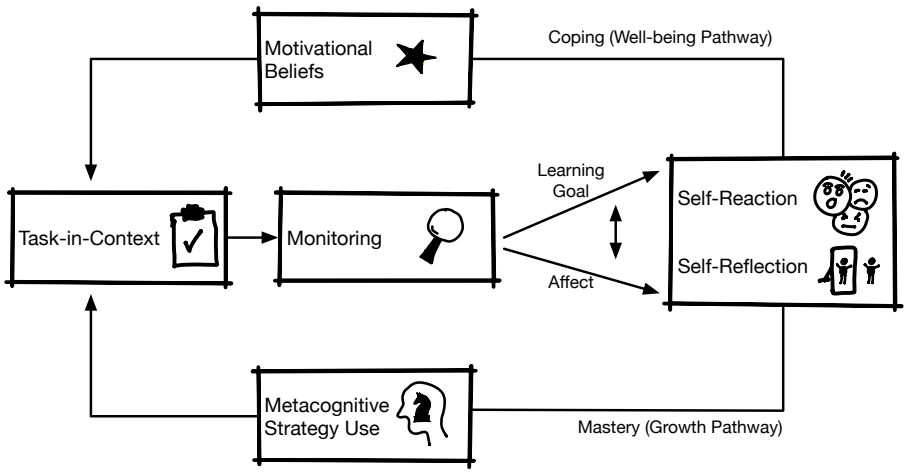

Fig. 7. Boekaerts Dual Processing Self-Regulation Model (2011)

The second (Figure 8) is Efklides' Metacognitive and Affective Model of Self-Regulated Learning [27]. This theory places more focus than the others on metacognitive knowledge rather than on the process of self-regulation. It include metacognitive knowledge about the person, the task, and, uniquely, the interaction of person-and-task. For regulating learning, it includes concurrent top-down and bottom-up forces. The task simply requires a certain set of knowledge and skills to complete, and it is treated like a context. The person-level is driven by the learner's goals for the task, providing top-down influence on regulation. It includes metacognitive knowledge, metacognitive skills, motivation, self-concept, and affect. The person-and-task level provides feedback to the learner based on performance and progress, providing bottom-up influence on regulation. These two forces ultimately dictate the learner's metacognitive knowledge and skills, motivation, self-concepts, and affect.

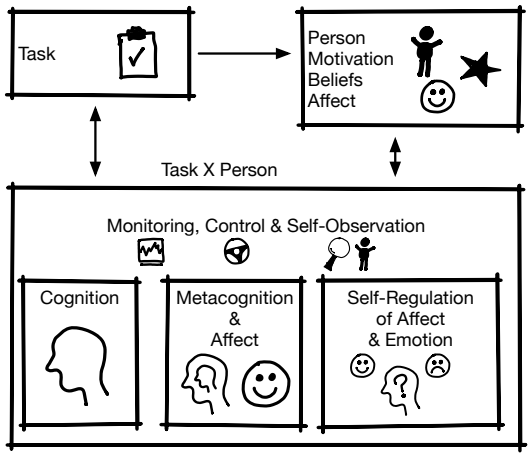

Fig. 8. Efklides Metacognitive and Affective Model of SRL (2011)

The third (Figure 9) is Winne & Hadwin's model of self-regulated learning [96]. Like Efklides' model, it also focuses on metacognition by constructing self-regulated learning as an information processing task. This task has four phases, like other self-regulated learning theories, but the first planning phase is split into two separate components. Most self-regulated learning models that build upon Bandura's self-regulation model [4] include a planning or forethought phase, but this is the only one to split it into task definition and then goal setting and planning. The other two phases, enacting strategies and metacognitive adaptation, are similar to other monitoring and self-reflection phases. This model becomes an information processing task by treating information from internal (i.e., the person) and external (i.e., task progress) as a feedback loop between the current state and goal state, making the task definition important enough to separate. This feedback loop includes five components of the information processing task of self-regulation: conditions (i.e., personal and environmental resources), operations (i.e., cognitive processes), products (i.e., information created through operations), evaluations (i.e., feedback about differences between products and standards), and standards (i.e., criteria for goal completion). These five components are represented as the COPES acronym and are carried forward to the last model we will discuss.

The last (Figure 10) is Hadwin et al.'s Socially Shared Regulated Learning (SSRL) model [38]. This model builds upon Winne & Hadwin's model [96] and COPES components to apply to collaborative learning contexts. Collaboration affects many of the common features interconnected with cognitive control because learners are managing their own and navigating others' cognition and motivation

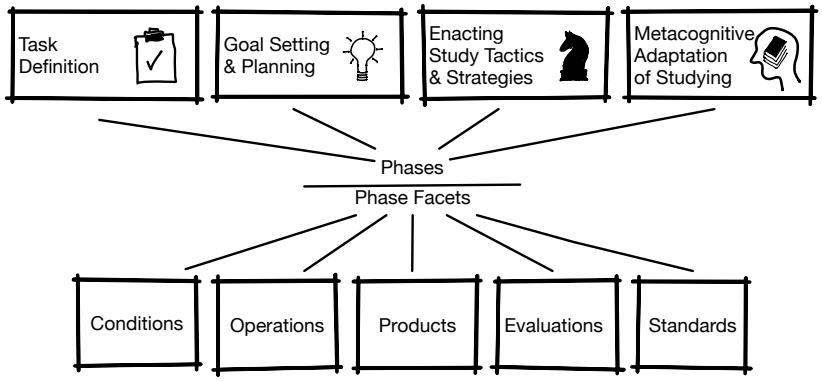

Fig. 9. Winne & Hadwin Information Processing SRL (1998)

through social interaction in the learning environment. Groups must coordinate planning, strategy use, and evaluation in addition to individual members regulating their own planning, strategy use, and evaluation. The SSRL model represents these interactions between self-regulation, co-regulation (supporting others' task regulation), and shared regulation of task strategies and progress towards goals. Essentially, the SSRL model is the same as Winne & Hadwin's model but carried out three times for self-regulation, co-regulation, and shared regulation. Just as in Winne & Hadwin, COPES components apply to each phase for each type of regulation.

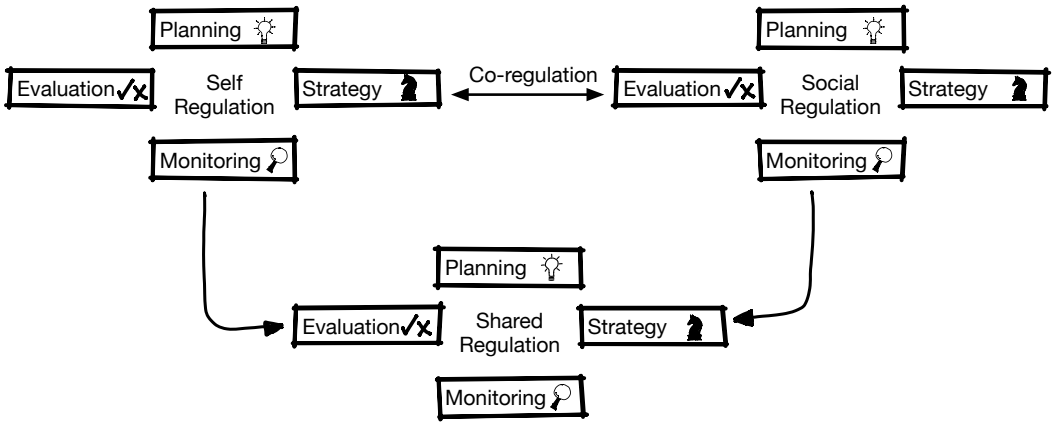

Fig. 10. Hadwin et al. Socially Shared Regulated Learning Model (2013)

7 CONCLUSIONS AND RECOMMENDATIONS

Computing education generally, and programming education specifically, could benefit from leveraging existing metacognitive and self-regulated learning theories that have been developed during the past several decades. Computing education researchers may benefit from using these theories to interpret their observations. This could lead to several developments including new ways to apply these theories, new ways to measure the effects of applying them, and new interpretive approaches specifically for use in computing education. Further, it is possible that computing-specific frameworks may be required in order to best apply metacognitive and self-regulated learning theories in

computing education contexts. It is possible that computing-specific theories could arise from such work.

The study of cognitive control becomes inextricably complex if researchers include all possible related constructs, such as self-efficacy, motivation, and social context. As a result, researchers must select which related constructs are most relevant to the aspect of cognitive control being studied. This need for selection has allowed multiple theories of cognitive control to thrive under different research contexts. Our goal for presenting various theories, including four that are not commonly used in computing education, is to aid this selection of theories. Our description of each theory includes which features make it unique from others. For example, if social constructs are important to the research question at hand, Hadwin et al.'s Socially Shared Regulated Learning model [38] is likely the best choice of theory because it is the only one to include a social dimension.

The descriptions of theories also include how they build upon and are presented in a way that is common to all descriptions. This includes details such as the diagrams of theories using shared icons to highlight which elements transcend more than one theory. We hope that this information helps readers understand the relationships between these theories and distinguish between them when deciding which is most appropriate for their specific context.

Recent work in computing education venues suggests that the computing education community sees potential in the use of metacognitive and self-regulated learning theory in interpreting observations in computing classrooms. This work should be of interest not just to computing education researchers but to practitioners who are interested in how theories that have proven useful in other disciplines may be applied in the context of computing education.

REFERENCES

[1] Satu Alaoutinen. 2012. Evaluating the Effect of Learning Style and Student Background on Self-assessment accuracy. *Computer Science Education* 22, 2 (2012), 175–198. https://doi.org/10.1080/08993408.2012.692924

[2] Kai Arakawa, Qiang Hao, Tyler Greer, Lu Ding, Christopher D. Hundhausen, and Abigayle Peterson. 2021. In Situ Identification of Student Self-Regulated Learning Struggles in Programming Assignments. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 467–473. https://doi.org/10.1145/3408877.3432357

[3] Carole A. Bagley and C. Candace Chou. 2007. Collaboration and the Importance for Novices in Learning Java Computer Programming. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Dundee, Scotland) *(ITiCSE '07)*. Association for Computing Machinery, New York, NY, USA, 211–215. https://doi.org/10.1145/1268784.1268846

[4] Albert Bandura. 1986. The Explanatory and Predictive Scope of Self-efficacy Theory. *Journal of Social and Clinical Psychology* 4, 3 (1986), 359–373.

[5] Albert Bandura. 1986. *Social Foundations of Thought and Action: A Social Cognitive Theory*. Prentice-Hall.

[6] Albert Bandura. 1997. *Self-efficacy: The Exercise of Control*. W H Freeman/Times Books/ Henry Holt & Co.

[7] Albert Bandura and Richard H. Walters. 1977. *Social Learning Theory*. Vol. 1. Prentice-Hall.

[8] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 177–210. https://doi.org/10.1145/3344429.3372508

[9] Susan Bergin and R. Reilly. 2005. The Influence of Motivation and Comfort-Level on Learning to Program. In *PPIG*. Psychology of Programming Interest Group, 293–304. https://ppig.org/papers/2005-ppig-17th-bergin/

[10] Susan Bergin, Ronan Reilly, and Desmond Traynor. 2005. Examining the Role of Self-Regulated Learning on Introductory Programming Performance. In *Proc. of the First International Workshop on Computing Education Research* (Seattle, WA, USA) *(ICER '05)*. ACM, New York, NY, USA, 81–86. https://doi.org/10.1145/1089786.1089794

[11] Katerine Bielaczyc, Peter L. Pirolli, and Ann L. Brown. 1995. Training in Self-Explanation and Self-Regulation Strategies: Investigating the Effects of Knowledge Acquisition Activities on Problem Solving. *Cognition and Instruction* 13, 2 (1995), 221–252. https://doi.org/10.1207/s1532690xci1302_3

[12] Benjamin Samuel Bloom. 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman.

[13] Monique Boekaerts and Lyn Corno. 2005. Self-Regulation in the Classroom: A Perspective on Assessment and Intervention. *Applied Psychology* 54, 2 (2005), 199–231. https://doi.org/10.1111/j.1464-0597.2005.00205.x

[14] J. Bransford, A.L. Brown, and R.R. Cocking. 2000. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. The National Academies Press, Washington, DC. https://doi.org/10.17226/9853

[15] Zack Butler, Ivona Bezakova, and Kimberly Fluet. 2017. Pencil Puzzles for Introductory Computer Science: An Experience- and Gender-Neutral Context. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 93–98. https://doi.org/10.1145/3017680.3017765

[16] Jennifer Campbell, Diane Horton, and Michelle Craig. 2016. Factors for Success in Online CS1. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (Arequipa, Peru) *(ITiCSE '16)*. Association for Computing Machinery, New York, NY, USA, 320–325. https://doi.org/10.1145/2899415.2899457

[17] Yuk Fai Cheong, Frank Pajares, and Paul S. Oberman. 2004. Motivation and Academic Help-Seeking in High School Computer Science. *Computer Science Education* 14, 1 (2004), 3–19. https://doi.org/10.1076/csed.14.1.3.23501

[18] Cheng-Yu Chung and I-Han Hsiao. 2020. Investigating Patterns of Study Persistence on Self-Assessment Platform of Programming Problem-Solving. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 162–168. https://doi.org/10.1145/3328778.3366827

[19] Tony Clear, JL Whalley, Phil Robbins, Anne Philpott, Anna Eckerdal, and Mikko-Jussi Laakso. 2011. Report on the final BRACElet workshop: Auckland University of Technology, September 2010. *Journal of Applied Computing and Information Technology* (2011).

[20] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46.

[21] Michelle Craig, Diane Horton, Daniel Zingaro, and Danny Heap. 2016. Introducing and Evaluating Exam Wrappers in CS2. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 285–290. https://doi.org/10.1145/2839509.2844561

[22] Michelle Craig, Andrew Petersen, and Jennifer Campbell. 2019. Answering the Correct Question. In *Proceedings of the ACM Conference on Global Computing Education* (Chengdu, Sichuan, China) *(CompEd '19)*. Association for Computing Machinery, New York, NY, USA, 72–77. https://doi.org/10.1145/3300115.3309529

[23] Kathryn Crawford and Alan Fekete. 1997. What Do Exam Results Really Measure?. In *Proceedings of the 2nd Australasian Conference on Computer Science Education* (The Univ. of Melbourne, Australia) *(ACSE '97)*. Association for Computing Machinery, New York, NY, USA, 185–190. https://doi.org/10.1145/299359.299386

[24] Paul Denny, James Prather, Brett A. Becker, Zachary Albrecht, Dastyni Loksa, and Raymond Pettit. 2019. A Closer Look at Metacognitive Scaffolding: Solving Test Cases Before Programming. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '19)*. Association for Computing Machinery, New York, NY, USA, Article 11, 10 pages. https://doi.org/10.1145/3364510.3366170

[25] Daniel L. Dinsmore, Patricia A. Alexander, and Sandra M. Loughlin. 2008. Focusing the Conceptual Lens on Metacognition, Self-regulation, and Self-regulated Learning. *Educational Psychology Review* 20, 4 (01 Dec 2008), 391–409. https://doi.org/10.1007/s10648-008-9083-6

[26] Shannon Duvall, Scott Spurlock, Dugald Ralph Hutchings, and Robert C. Duvall. 2021. Improving Content Learning and Student Perceptions in CS1 with Scrumage. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 474–480. https://doi.org/10.1145/3408877.3432415

[27] Anastasia Efklides. 2008. Metacognition: Defining its Facets and Levels of Functioning in Relation to Self-regulation and Co-regulation. *European Psychologist* 13, 4 (2008), 277–287.

[28] Peggy A. Ertmer and Timothy J. Newby. 1996. Expert Learner: Strategic, Self-regulated, and Reflective. *Instructional Science* 24, 1 (1996), 1–24.

[29] Anneli Eteläpelto. 1993. Metacognition and the Expertise of Computer Program Comprehension. *Scandinavian Journal of Educational Research* 37, 3 (1993), 243–254. https://doi.org/10.1080/0031383930370305

[30] Katrina Falkner, Claudia Szabo, Rebecca Vivian, and Nickolas Falkner. 2015. Evolution of Software Development Strategies. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2* (Florence, Italy) *(ICSE '15)*. IEEE Press, 243–252.

[31] Katrina Falkner, Rebecca Vivian, and Nickolas J.G. Falkner. 2014. Identifying Computer Science Self-Regulated Learning Strategies. In *Proc. of the 2014 Conference on Innovation and Technology in Computer Science Education* (Uppsala, Sweden) *(ITiCSE '14)*. ACM, New York, NY, USA, 291–296. https://doi.org/10.1145/2591708.2591715

[32] Nickolas Falkner, Rebecca Vivian, David Piper, and Katrina Falkner. 2014. Increasing the Effectiveness of Automated Assessment by Increasing Marking Granularity and Feedback Units. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 9–14. https://doi.org/10.1145/2538862.2538896

[33] J. H. Flavell. 1976. Metacognitive Aspects of Problem Solving. *The Nature of Intelligence* (1976), 231–235. https://ci.nii.ac.jp/naid/10021876052/en/

[34] John H Flavell. 1979. Metacognition and Cognitive Monitoring: A New Area of Cognitive-Developmental Inquiry. *American psychologist* 34, 10 (1979), 906.

[35] Emily Fox and Michelle Riconscente. 2008. Metacognition and Self-regulation in James, Piaget, and Vygotsky. *Educational Psychology Review* 20, 4 (2008), 373–389.

[36] 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 *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 91–101. https://doi.org/10.1145/3372782.3406257

[37] Jamie Gorson and Eleanor O'Rourke. 2020. Why Do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 170–181. https://doi.org/10.1145/3372782.3406273

[38] Allyson Fiona Hadwin, Sanna Järvelä, and Mariel Miller. 2018. Self-regulated, Co-regulated, and Socially Shared Regulation of Learning. In *Handbook of Self-regulation of Learning and Performance*, D. H. Schunk and J. A. Greene Greene (Eds.). Vol. 30. Routledge/Taylor & Francis Group, 83–106.

[39] Alison Hull. 2014. *Motivational and Metacognitive Feedback in an ITS: Linking Past States and Experiences to Current Problems.* Ph.D. Dissertation. University of Sussex.

[40] Alison Hull and Benedict du Boulay. 2015. Motivational and Metacognitive Feedback in SQL-Tutor*. *Computer Science Education* 25, 2 (2015), 238–256. https://doi.org/10.1080/08993408.2015.1033143

[41] Kalle Ilves, Juho Leinonen, and Arto Hellas. 2018. Supporting Self-Regulated Learning with Visualizations in Online Learning Environments. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 257–262. https://doi.org/10.1145/3159450.3159509

[42] Ville Isomöttönen and Ville Tirronen. 2016. Flipping and Blending—An Action Research Project on Improving a Functional Programming Course. *ACM Trans. Comput. Educ.* 17, 1, Article 1 (Sept. 2016), 35 pages. https://doi.org/10.1145/2934697

[43] Shekhar Kalra, Charles Thevathayan, and Margaret Hamilton. 2020. Developing Industry-Relevant Higher Order Thinking Skills in Computing Students. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 294–299. https://doi.org/10.1145/3341525.3387381

[44] Viggo Kann and Anna-Karin Högfeldt. 2016. Effects of a Program Integrating Course for Students of Computer Science and Engineering. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 510–515. https://doi.org/10.1145/2839509.2844610

[45] Amanpreet Kapoor and Christina Gardner-McCune. 2020. Exploring the Participation of CS Undergraduate Students in Industry Internships. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1103–1109. https://doi.org/10.1145/3328778.3366844

[46] Ada S. Kim and Amy J. Ko. 2017. A Pedagogical Analysis of Online Coding Tutorials. In *Proc. of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. ACM, New York, NY, USA, 321–326. https://doi.org/10.1145/3017680.3017728

[47] Michael S. Kirkpatrick, Mohamed Aboutabl, David Bernstein, and Sharon Simmons. 2015. Backward Design: An Integrated Approach to a Systems Curriculum. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (Kansas City, Missouri, USA) *(SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 30–35. https://doi.org/10.1145/2676723.2677264

[48] Michael S. Kirkpatrick and Samantha Prins. 2015. Using the Readiness Assurance Process and Metacognition in an Operating Systems Course. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15)*. Association for Computing Machinery, New York, NY, USA, 183–188. https://doi.org/10.1145/2729094.2742594

[49] Sophia Krause-Levy, Leo Porter, Beth Simon, and Christine Alvarado. 2020. Investigating the Impact of Employing Multiple Interventions in a CS1 Course. In *Proceedings of the 51st ACM Technical Symposium on Computer Science*

Education (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1082–1088. https://doi.org/10.1145/3328778.3366866

[50] Einari Kurvinen, Rolf Lindén, Teemu Rajala, Erkki Kaila, Mikko-Jussi Laakso, and Tapio Salakoski. 2012. Computer-Assisted Learning in Primary School Mathematics Using ViLLE Education Tool. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '12)*. Association for Computing Machinery, New York, NY, USA, 39–46. https://doi.org/10.1145/2401796.2401801

[51] Priscilla Lee and Soohyun Nam Liao. 2021. Targeting Metacognition by Incorporating Student-Reported Confidence Estimates on Self-Assessment Quizzes. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 431–437. https://doi.org/10.1145/3408877.3432377

[52] Leo Leppänen, Juho Leinonen, and Arto Hellas. 2016. Pauses and Spacing in Learning to Program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '16)*. Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/2999541.2999549

[53] 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 *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. Association for Computing Machinery, New York, NY, USA, 211–220. https://doi.org/10.1145/2960310.2960329

[54] Dastyni Loksa and Amy J. Ko. 2016. The Role of Self-Regulation in Programming Problem Solving Process and Success. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. Association for Computing Machinery, New York, NY, USA, 83–91. https://doi.org/10.1145/2960310.2960334

[55] Dastyni Loksa, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. 2016. Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 1449–1461. https://doi.org/10.1145/2858036.2858252

[56] Dastyni Loksa, Benjamin Xie, Harrison Kwik, and Amy J. Ko. 2020. Investigating Novices' In Situ Reflections on Their Programming Process. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 149–155. https://doi.org/10.1145/3328778.3366846

[57] Lauri Malmi, Judy Sheard, Päivi Kinnunen, Simon, and Jane Sinclair. 2019. Computing Education Theories: What Are They and How Are They Used?. In *Proc. of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. ACM, New York, NY, USA, 187–197. https://doi.org/10.1145/3291279.3339409

[58] Lauri Malmi, Judy Sheard, Päivi Kinnunen, and Jane Sinclair. 2020. Theories and models of emotions, attitudes, and self-efficacy in the context of programming education. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. 36–47.

[59] Murali Mani and Quamrul Mazumder. 2013. Incorporating Metacognition into Learning. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 53–58. https://doi.org/10.1145/2445196.2445218

[60] Joshua Martin, Stephen H. Edwards, and Clfford A. Shaffer. 2015. The Effects of Procrastination Interventions on Programming Project Success. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (Omaha, Nebraska, USA) *(ICER '15)*. Association for Computing Machinery, New York, NY, USA, 3–11. https://doi.org/10.1145/2787622.2787730

[61] Samiha Marwan, Anay Dombe, and Thomas W. Price. 2020. Unproductive Help-Seeking in Programming: What It is and How to Address It. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 54–60. https://doi.org/10.1145/3341525.3387394

[62] Adon Christian Michael Moskal and Rob Wass. 2019. Interpersonal Process Recall: A Novel Approach to Illuminating Students' Software Development Processes. *Computer Science Education* 29, 1 (2019), 5–22. https://doi.org/10.1080/08993408.2018.1542190

[63] Erica Mourão, João Felipe Pimentel, Leonardo Murta, Marcos Kalinowski, Emilia Mendes, and Claes Wohlin. 2020. On the Performance of Hybrid Search Strategies for Systematic Literature Reviews in Software Engineering. *Information and Software Technology* 123 (2020), 106294. https://doi.org/10.1016/j.infsof.2020.106294

[64] Laurie Murphy and Josh Tenenberg. 2005. Do Computer Science Students Know What They Know? A Calibration Study of Data Structure Knowledge. (2005), 148–152. https://doi.org/10.1145/1067445.1067488

[65] Claudia Ott, Anthony Robins, Patricia Haden, and Kerry Shephard. 2015. Illustrating performance indicators and course characteristics to support students' self-regulated learning in CS1. *Computer Science Education* 25, 2 (2015), 174–198. https://doi.org/10.1080/08993408.2015.1033129

[66] F. Pajares. 1997. Current Directions in Self-efficacy Research. In *Advances in Motivation and Achievement*, M. Maehr and P. R. Pintrich (Eds.). JAI Press, Greenwich, CT, 1–49.

[67] Ernesto Panadero. 2017. A Review of Self-regulated Learning: Six Models and Four Directions for Research. *Frontiers in Psychology* 8 (2017), 422. https://doi.org/10.3389/fpsyg.2017.00422

[68] Jennifer Parham, Leo Gugerty, and D. E. Stevenson. 2010. Empirical Evidence for the Existence and Uses of Metacognition in Computer Science Problem Solving. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (Milwaukee, Wisconsin, USA) *(SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 416–420. https://doi.org/10.1145/1734263.1734406

[69] S. G. Paris and P. Winograd. 1990. How Metacognition Can Promote Academic Learning and Instruction. In *Dimensions of Thinking and Cognitive Instruction*, B. F. Jones and L. Idol (Eds.). Lawrence Erlbaum Associates, Inc., 15–51.

[70] Miranda C. Parker, Kantwon Rogers, Barbara J. Ericson, and Mark Guzdial. 2017. Students and Teachers Use An Online AP CS Principles EBook Differently: Teacher Behavior Consistent with Expert Learners. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (Tacoma, Washington, USA) *(ICER '17)*. Association for Computing Machinery, New York, NY, USA, 101–109. https://doi.org/10.1145/3105726.3106189

[71] Markeya S. Peteranetz, Patrick M. Morrow, and Leen-Kiat Soh. 2020. Development and Validation of the Computational Thinking Concepts and Skills Test. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 926–932. https://doi.org/10.1145/3328778.3366813

[72] Markeya S. Peteranetz, Shiyuan Wang, Duane F. Shell, Abraham E. Flanigan, and Leen-Kiat Soh. 2018. Examining the Impact of Computational Creativity Exercises on College Computer Science Students' Learning, Achievement, Self-Efficacy, and Creativity. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 155–160. https://doi.org/10.1145/3159450.3159459

[73] Paul R. Pintrich. 2000. Chapter 14 - The Role of Goal Orientation in Self-Regulated Learning. In *Handbook of Self-Regulation*, Monique Boekaerts, Paul R. Pintrich, and Moshe Zeidner (Eds.). Academic Press, San Diego, 451 – 502. https://doi.org/10.1016/B978-012109890-2/50043-3

[74] Paul R Pintrich and Elisabeth V De Groot. 1990. Motivational and Self-regulated Learning Components of Classroom Academic Performance. *Journal of Educational Psychology* 82, 1 (1990), 33–40.

[75] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. 2020. What Do We Think We Think We Are Doing? Metacognition and Self-Regulation in Programming. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 2–13. https://doi.org/10.1145/3372782.3406263

[76] James Prather, Raymond Pettit, Brett A. Becker, Paul Denny, Dastyni Loksa, Alani Peters, Zachary Albrecht, and Krista Masci. 2019. First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 531–537. https://doi.org/10.1145/3287324.3287374

[77] James Prather, Raymond Pettit, Kayla McMurry, Alani Peters, John Homer, and Maxine Cohen. 2018. Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (Espoo, Finland) *(ICER '18)*. Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/3230977.3230981

[78] James Prather, Raymond Pettit, Kayla Holcomb McMurry, Alani Peters, John Homer, Nevan Simone, and Maxine Cohen. 2017. On Novices' Interaction with Compiler Error Messages: A Human Factors Approach. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (Tacoma, Washington, USA) *(ICER '17)*. Association for Computing Machinery, New York, NY, USA, 74–82. https://doi.org/10.1145/3105726.3106169

[79] V.G. Renumol, Dharanipragada Janakiram, and S. Jayaprakash. 2010. Identification of Cognitive Processes of Effective and Ineffective Students During Computer Programming. *ACM Trans. Comput. Educ.* 10, 3, Article 10 (Aug. 2010), 21 pages. https://doi.org/10.1145/1821996.1821998

[80] Alexander Ruf, Andreas Mühling, and Peter Hubwieser. 2014. Scratch vs. Karel: Impact on Learning Outcomes and Motivation. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (Berlin, Germany) *(WiPSCE '14)*. Association for Computing Machinery, New York, NY, USA, 50–59. https://doi.org/10.1145/2670757.2670772

[81] Jean Salac, Cathy Thomas, Chloe Butler, and Diana Franklin. 2021. Supporting Diverse Learners in K-8 Computational Thinking with TIPP&SEE. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 246–252. https://doi.org/10.1145/3408877.3432366

[82] Phil Sands and Aman Yadav. 2020. Self-Regulation for High School Learners in a MOOC Computer Science Course. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*.

Association for Computing Machinery, New York, NY, USA, 845–851. https://doi.org/10.1145/3328778.3366818

[83] Dale H Schunk. 2008. Metacognition, Self-regulation, and Self-regulated Learning: Research Recommendations. *Educational Psychology Review* 20, 4 (2008), 463–467.

[84] Dale H. Schunk, Paul R. Pintrich, and Judith L. Meece. 2008. *Motivation in Education: Theory, Research, and Applications*. Pearson Merrill Prentice Hall, Upper Saddle River, NJ.

[85] Clifford A. Shaffer and Ayaan M. Kazerouni. 2021. The Impact of Programming Project Milestones on Procrastination, Project Outcomes, and Course Outcomes: A Quasi-Experimental Study in a Third-Year Data Structures Course. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 907–913. https://doi.org/10.1145/3408877.3432356

[86] Leonardo Silva, António José Mendes, Anabela Gomes, and Gabriel Fortes Cavalcanti de Macêdo. 2021. Regulation of Learning Interventions in Programming Education: A Systematic Literature Review and Guideline Proposition. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 647–653. https://doi.org/10.1145/3408877.3432363

[87] Ben Stephenson, Michelle Craig, Daniel Zingaro, Diane Horton, Danny Heap, and Elaine Huynh. 2017. Exam Wrappers: Not a Silver Bullet. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 573–578. https://doi.org/10.1145/3017680.3017701

[88] Robert J Sternberg et al. 1985. *Beyond IQ: A Triarchic Theory of Human Intelligence*. CUP Archive.

[89] Jeffrey A. Stone and Elinor M. Madigan. 2007. Integrating reflective writing in CS/IS. *ACM SIGCSE Bulletin* 39, 2 (jun 2007), 42. https://doi.org/10.1145/1272848.1272881

[90] 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 *Proc. of the WG Reports on Innovation and Technology in Comp Sci Education* (Aberdeen, Scotland Uk) *(ITiCSE-WGR '19)*. ACM, NY, USA, 89–109. https://doi.org/10.1145/3344429.3372504

[91] Josh Tenenberg and Laurie Murphy. 2005. Knowing What I Know: An Investigation of Undergraduate Knowledge and Self-knowledge of Data Structures. *Computer Science Education* 15, 4 (2005), 297–315. https://doi.org/10.1080/08993400500307677

[92] Tammy VanDeGrift, Tamara Caruso, Natalie Hill, and Beth Simon. 2011. Experience Report: Getting Novice Programmers to THINK about Improving Their Software Development Process. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 493–498. https://doi.org/10.1145/1953163.1953307

[93] Arto Vihavainen, Craig S. Miller, and Amber Settle. 2015. Benefits of Self-Explanation in Introductory Programming. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (Kansas City, Missouri, USA) *(SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 284–289. https://doi.org/10.1145/2676723.2677260

[94] Rebecca Vivian, Katrina Falkner, Nickolas Falkner, and Hamid Tarmazdi. 2016. A Method to Analyze Computer Science Students' Teamwork in Online Collaborative Learning Environments. *ACM Trans. Comput. Educ.* 16, 2, Article 7 (Feb. 2016), 28 pages. https://doi.org/10.1145/2793507

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

[96] Philip H Winne and Allyson F Hadwin. 1998. Studying as Self-Regulated Engagement in Learning. *Metacognition in Educational Theory and Practice* (1998), 277–304.

[97] 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. *Computer Science Education* 29, 2-3 (2019), 205–253. https://doi.org/10.1080/08993408.2019.1565235

[98] Aman Yadav, Chris Mayfield, Sukanya Kannan Moudgalya, Clif Kussmaul, and Helen H. Hu. 2021. Collaborative Learning, Self-Efficacy, and Student Performance in CS1 POGIL. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 775–781. https://doi.org/10.1145/3408877.3432373

[99] Lisa Yan, Annie Hu, and Chris Piech. 2019. Pensieve: Feedback on Coding Process for Novices. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 253–259. https://doi.org/10.1145/3287324.3287483

[100] Barry J Zimmerman. 1989. A Social Cognitive View of Self-regulated Academic Learning. *Journal of Educational Psychology* 81, 3 (1989), 329.

[101] Barry J. Zimmerman. 1990. Self-Regulated Learning and Academic Achievement: An Overview. *Educational Psychologist* 25, 1 (1990), 3–17. https://doi.org/10.1207/s15326985ep2501_2

[102] Barry J. Zimmerman. 2000. Chapter 2 - Attaining Self-Regulation: A Social Cognitive Perspective. In *Handbook of Self-Regulation*, Monique Boekaerts, Paul R. Pintrich, and Moshe Zeidner (Eds.). Academic Press, San Diego, 13 – 39. https://doi.org/10.1016/B978-012109890-2/50031-7

A APPENDIX

Author	Ref	Title	Year
Crawford & Fekete	[23]	What Do Exam Results Really Measure?	1997
Bergin et al.	[10]	Examining the Role of Self-regulated Learning on Introductory Programming Performance	2005
Murphy & Tenenberg	[64]	Do Computer Science Students Know What They Know?: A Calibration Study of Data Structure Knowledge	2005
Bagley & Chou	[3]	Collaboration and the Importance for Novices in Learning Java Computer Programming	2007
Stone & Madigan	[89]	Integrating Reflective Writing in CS/IS	2007
Parham et al.	[68]	Empirical Evidence for the Existence and Uses of Metacognition in Computer Science Problem Solving	2010
VanDeGrift et al.	[92]	Experience Report: Getting Novice Programmers to THINK About Improving Their Software Development Process	2011
Kurvinen et al.	[50]	Computer-assisted Learning in Primary School Mathematics Using ViLLE Education Tool	2012
Mani & Mazumder	[59]	Incorporating Metacognition into Learning	2013
Falkner et al.	[31]	Identifying Computer Science Self-regulated Learning Strategies	2014
Ruf et al.	[80]	Scratch vs. Karel: Impact on Learning Outcomes & Motivation	2014
Watson et al.	[95]	No Tests Required: Comparing Traditional & Dynamic Predictors of Programming Success	2014
Kirkpatrick et al.	[47]	Backward Design: An Integrated Approach to a Systems Curriculum	2015
Kirkpatrick et al.	[48]	Using the Readiness Assurance Process & Metacognition in an Operating Systems Course	2015
Martin et al.	[60]	The Effects of Procrastination Interventions on Programming Project Success	2015
Vihavainen et al.	[93]	Benefits of Self-explanation in Introductory Programming	2015
Campbell et al.	[16]	Factors for Success in Online CS1	2016
Craig et al.	[21]	Introducing & Evaluating Exam Wrappers in CS2	2016
Kann & Högfeldt	[44]	Effects of a Program Integrating Course for Students of Computer Science and Engineering	2016
Leppänen et al.	[52]	Pauses and Spacing in Learning to Program	2016
Lishinski et al.	[53]	Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance	2016
Loksa & Ko	[54]	The Role of Self-Regulation in Programming Problem Solving Process & Success	2016
Butler et al.	[15]	Pencil Puzzles for Introductory Computer Science: An Experience- and Gender-Neutral Context	2017
Kim & Ko	[46]	A Pedagogical Analysis of Online Coding Tutorials	2017
Parker et al.	[70]	Students and Teachers Use An Online AP CS Principles EBook Differently: Teacher Behavior Consistent with Expert Learners	2017
Stephenson et al.	[87]	Exam Wrappers: Not a Silver Bullet	2017
Ilves et al.	[41]	Supporting Self-Regulated Learning with Visualizations in Online Learning Environments	2018
Peteranetz et al.	[72]	Examining the Impact of Computational Creativity Exercises on College Computer Science Students' Learning, Achievement, Self-Efficacy, and Creativity	2018
Prather et al.	[77]	Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools	2018
Prather et al.	[76]	First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts	2019
Yan et al.	[99]	Pensieve: Feedback on Coding Process for Novices	2019

Table 1. *Depth* papers found by Prather et al. [75] from conferences (ACM DL sponsor: SIGCSE) through 26/11/2019

Author	Ref	Title	Year
Cheong et al.	[17]	Motivation & Academic Help-Seeking in High School Computer Science	2004
Tenenberg & Murphy	[91]	Knowing What I Know: An investigation of Undergraduate Knowledge and self-knowledge of data structures	2005
Renumol et al.	[79]	Identification of Cognitive Processes of Effective and Ineffective Students During Computer Programming	2010
Alaoutinen	[1]	Evaluating the Effect of Learning Style and Student Background on Self-assessment accuracy	2012
Hull & du Boulay	[40]	Motivational and Metacognitive Feedback in SQL-Tutor*	2015
Ott et al.	[65]	Illustrating Performance Indicators and Course Characteristics to Support Students' self-regulated learning in CS2	2015
Isomöttönen & Tirronen	[42]	Flipping and Blending—An Action Research Project on Improving a Functional Programming Course	2016
Vivian et al.	[94]	A Method to Analyze Computer Science Students' Teamwork in Online Collaborative Learning Environments	2016
Moskal & Wass	[62]	Interpersonal Process Recall: a Novel Approach to Illuminating Students' Software development processes	2019
Xie et al.	[97]	A Theory of Instruction for Introductory Programming Skills	2019
Chung & Hsiao	[18]	Investigating Patterns of Study Persistence on Self-Assessment Platform of Programming Problem-Solving	2020
Franklin et al.	[36]	Exploring Student Behavior Using the TIPP&SEE Learning Strategy	2020
Gorson & O'Rourke	[37]	Why Do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities	2020
Kalra et al.	[43]	Developing Industry-Relevant Higher Order Thinking Skills in Computing Students	2020
Kapoor & Gardner-McCune	[45]	Exploring the Participation of CS Undergraduate Students in Industry Internships	2020
Krause-Levy et al.	[49]	Investigating the Impact of Employing Multiple Interventions in a CS1 Course	2020
Loksa et al.	[56]	Investigating Novices' In Situ Reflections on Their Programming Process	2020
Marwan et al.	[61]	Unproductive Help-Seeking in Programming: What It is and How to Address It	2020
Peteranetz et al.	[71]	Development and Validation of the Computational Thinking Concepts and Skills Test	2020
Sandsl & Yadav	[82]	Self-Regulation for High School Learners in a MOOC Computer Science Course	2020
Arakawa et al.	[2]	In Situ Identification of Student Self-Regulated Learning Struggles in Programming Assignments	2021
Duvall et al.	[26]	Improving Content Learning and Student Perceptions in CS1 with Scrumage	2021
Lee & Liao	[51]	Targeting Metacognition by Incorporating Student-Reported Confidence Estimates on Self-Assessment Quizzes	2021
Salac et al.	[81]	Supporting Diverse Learners in K-8 Computational Thinking with TIPP & SEE	2021
Shaffer & Kazerouni	[85]	The Impact of Programming Project Milestones on Procrastination, Project Outcomes, and Course Outcomes: A Quasi-Experimental Study in a Third-Year Data Structures Course	2021
Silva et al.	[86]	Regulation of Learning Interventions in Programming Education: A Systematic Literature Review and Guideline Proposition	2021
Yadav et al.	[98]	Collaborative Learning, Self-Efficacy, and Student Performance in CS1 POGIL	2021

Table 2. *Depth* papers found by this study from conferences (ACM DL sponsor: SIGCSE) and *TOCE*, *Computer Science Education* through 21/3/2021

Author	Ref	Title	Year
Eteläpelto	[29]	Metacognition and the Expertise of Computer Program Comprehension	1993
Bielaczyc et al.	[11]	Training in Self-explanation and Self-regulation Strategies: Investigating the Effects of Knowledge Acquisition Activities on Problem Solving	1995
Bergin & Reilly	[9]	The Influence of Motivation and Comfort-level on Learning to Program	2005
Falkner et al.	[30]	Evolution of Software Development Strategies	2015
Loksa et al.	[55]	Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance	2016

Table 3. *Depth* papers found by this study from reverse-snowball search using papers from Prather et al. [75] and this study (Tables 1 and 2)