Self-Regulated Learning Skills Research in Computer Science: The State of the Field

Molly Domino

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Clifford A.Shaffer, Chair
Stephen H. Edwards
Alan C. Jamieson
Sara Hooshangi
Brett D. Jones
Kenneth R. Edmison, Jr.

June 28, 2024
Blacksburg, Virginia

Keywords: Self-Regulated Learning, SRL, Learning Analytics, Data Traces, Metacognition,
Unobtrusive Measures,

Self-Regulated Learning Skills Research in Computer Science: The State of the Field

Molly Domino

(ABSTRACT)

Academic success requires not only taking in content, but also understanding how to learn best. Self Regulated Learning (SRL) is process by which humans regulate their thinking, emotions, and behavior. It broadly describes the process of knowing (or learning) how to learn. Education research has found Self-Regulated Learning to be a key predictor of academic success along with other constructs like motivation and self-efficacy. It may be particularly critical in learning to program at the post-secondary level. Studies have shown that students benefit greatly from targeted instruction in these skills. Teaching students how to better self-regulate is both important and valuable for Computer Science students. The solution here may seem straightforward: educators should give instruction on self-regulation skills. However, there are a number of skills that encompass a student's proficiency with self-regulate; including time management, problem decomposition, and reflection. Self regulation also tends to be a highly cognitive and internal process making it difficult to observe directly, let alone measure. Which skills should be prioritized for targeted instruction?

How could we empirically measure those skills? What limitations should we keep in mind when making such decisions? Within this dissertation, I will seek to address these questions. In order to get an idea of what skills the Computing Education Research community should be prioritizing, my co-authors and I conducted two studies. First, a Delphi Process study that expanded the field by gaining an understanding of what SRL skills CS post-secondary

educators value most. This gave a more firm view of what skills were most important for CS students. Second, a systematic literature review to examine what skills had been studied within the Computing Education Research community. Ultimately, I created a finalized list of 12 SRL skills that appear to be particularly important to CS education. This list also includes behaviors an outside observer could use as indicators of the presence or absence of SRL.

After creating this list, I then considered how best to measure these each of these 12 skills. One form of measurement comes from using data traces collected from educational software. These allow researchers to make strong inferences about a student's internal state empirically. They also allow for measurement of students at greater scale and through automated means, making them advantageous for large classes. For my third publication, I then set about identifying a set of data traces for these skills taking a theory-first approach. I also make the case that CS is well situated to make great gains in trace-based approaches as they make use of a whole ecosystem of data sources. This is important as it is currently common for studies to utilize just one.

Self-Regulated Learning Skills Research in Computer Science: The State of the Field

Molly Domino

(GENERAL AUDIENCE ABSTRACT)

Knowing how to learn is a critical aspect to academic success. Self-Regulation is the process by which humans regulate their thinking, emotions, and behavior. It encompasses the process of knowing (or learning) how to learn. Several studies have argued that learning Computer Science especially requires a strong self-regulated learning, but studies show novice programmer's skills in this area are still weak and benefit from further instruction. This is true even for students entering post-secondary education. Thus teaching students how to better self-regulate is important for CS students, but creating such lessons is not straightforward. SRL is a broad field and covers a variety of different skills that students may need. What skills are most important for instructors to teach their students? Once we know what skills are most important for targeting, how do we measure those skills? These are the questions I examine.

In order to get an idea of what skills the Computing Education Research community should be prioritizing, I conducted both a Delphi Process study. Following that I conducted a systematic literature review to get a better idea of what the Computing Education Research community is currently studying. I then considered the best way to measure these skills. While there are many approaches available to study SRL, I opted to examine these skills through student interactions with digital education software, called data traces. These traces are advantageous as they authentically capture learning in a way no other approach currently can. For my third paper I systematically derived a series of high-quality traces

and made the case that CS classes already collect a lot of valuable traces through common digital education software systems.

Dedication

To Conor Wallace, Sylvia Pabreza and Frank Domino.

Thank you for your endless patience and encouragement.

Acknowledgments

So very many people have helped me along this journey of pursuing my PhD. It feels a little odd to say that 'I did this' when so many other people contributed. I can only directly thank a few of you but please know, if you are reading this acknowledgements section, you did so much to help me and I am tremendously grateful. To all of the friends and mentors who have supported me through this process, whether you are named here or not, thank you for inspiring me to continue and getting me through the highs and lows of this process.

First, I owe particular thanks to my committee for their invaluable guidance and support. I am very grateful to Dr.Clifford Shaffer for his patience, attention to detail, and unending ability to bring out the best in me. His feedback every step of the way have shaped so much of my work and myself as a whole through graduate school. I am also indebted to Dr.Alan Jamieson for his unparalleled mentorship and endless belief in my abilities. The conversations I had with Dr.Steve Edwards always left me with a real drive and curiosity to to examine the world more richly and deeply. I'd like to thank Dr.Bob Edmison for his invaluable support over the last few years. Dr.Brett Jones and Dr.Sara Hooshangi, thank you for taking the time to guide me through this process and offer your feedback. I'd also like to note, that everyone on my committee went out of their way to meet with me and make this dissertation happen. To all my committee, thank you so much for being so accommodating and supportive throughout my career at Virginia Tech.

While not on my committee, I would also like to extend thanks to Margaret Ellis and Dr.Dennis Kafura.Both encouraged me to pursue this degree and were wonderful mentors

early on in my graduate school career. To my labmates, Arinjoy Basak, Rifat Sabbir Mansur, Alex Hicks, and Alexandra Thompson, thank you for making my days a little brighter. Thank you also to Dr.Lindsay Jamieson, who took her free time out her very busy life to read and offer edits on my writing for no other reason than she is a kind and wonderful person. Both Dr.s Jamiesons have been once in a lifetime mentors and friends. Thank you also to everyone at the Roux Institute at Northeastern University for welcoming me in to your community and for offering guidance on how to be both a better researcher and a better educator. Special thanks also to Gary Cantrell, Brianna Dym, and Ryan Bockman for your help and encouragement to both become a better researcher and teacher as I progressed toward the finish line.

To my parents, Sylvia Pabreza and Frank Domino, whose unconditional love made this work possible. Your endless patience, love, and support with me as I work never ceases to amaze me. Thank you so much. To Conor Wallace, my partner, who has never once seemed to doubt that I could get this far (even when I did myself), I cannot thank enough. You have not just supported me, but somehow managed to encourage me to do my best work while also making sure that I take time care for myself as well. Your support from the day I started to this final push over the finish line has been so helpful and has made this process infinitely better. These last 5 years would have been unfathomable without you next to me every day. Lastly, thank you to the other members of my household, Willow and Ash, who tried their best; even if they like stepping on keyboards too much to really contribute.

Contents

List of Figures

List of Tables

List of Definitions

CER Computing Education Research

Data Source The digital education software tools that students engage to generate data traces

Data Trace a bit of ambient data that affords relatively strong inferences about one or more cognitive, affective, metacognitive, and motivational states and processes [158]

Learning Analytics (LA) The measurement, collection, analysis, and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs [43]

Self-Regulation Behavior An observable action or evidence that indicates the presence (or absence) of a particular SRL skill

Self-Regulation Skill An internal mastery of a particular aspect of self regulation

SRL Self-Regulated Learning

Trace A datum or dataset gathered from student interactions with educational software in order to make an inference about a student's cognitive state

Chapter 1

Introduction

Academic success requires not only knowing what to learn but also how to learn it. The process by which humans regulate their thinking, emotions, and behavior towards learning goals is called "Self-Regulated Learning", which broadly describes the process of learning how to learn. There is growing interest within the Computing Education Research (CER) community to better understand how these constructs are used when learning to program.

While Self-Regulated Learning (SRL) is often studied in primary and secondary education, studies show novice programmers' proficiency with self regulation is strengthened by interventions— even at the post-secondary level. Thus, teaching students how to better regulate their learning is both important and valuable for post-secondary Computer Science students.

However, there is ultimately a lot of nuance that goes into deciding what aspects of self regulation to focus on. Even then, skills need iterative cycles of practice and feedback. Such cycles cannot be assessed unless there are measures with which to watch that skill develop.

With that in mind, this dissertation is focused on addressing the following two research questions:

- **RQ1:** What self-regulation skills should be prioritized for intervention within a CS classroom?

- **RQ2:** How do we measure those skills most effectively?

The major contribution of this dissertation is to identify a shortlist of self-regulation skills and measures for those skills. Specifically, I discuss data traces that can be used to empirically measure self-regulation within the CS classroom.

I chose to focus on identifying data traces as they offer an opportunity to examine this educational construct unobtrusively and can be used in large classes (see Section 2.2 for more detail).

In order to answer Research Question 1, I needed to get an idea of what skills the CER community should be prioritizing. I (and my co-authors) conducted both a Delphi Process study and a systematic literature review to create a list of 12 self-regulation skills that appear to be particularly important to CS education. This list also includes a shortlist of behaviors an outside observer could use as indicators of the presence or absence of self regulation.

Once I had a list of skills to prioritize, I then turned to considering how future researchers and educators could assess the efficacy of any potential intervention. For this research, my goal was to identify traces that any future researcher could use to identify self regulation in action. With that in mind, I prioritized having traces that were Due to the fact that data traces have the opportunity to observe learning in a more authentic way than other approaches, I opted to focus on measures that could be taken empirically and unobtrusively. Thus, I began to examine how the field of Learning Analytics (LA) was measuring SRL. The field of LA focuses on: "the measurement, collection, analysis, and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs" [43]. Over the past few decades, researchers in this domain have begun to leverage interaction data from digital education software to empirically and unobtrusively measure how students self-regulate their learning.

We call such interaction data a 'data trace', or: "...a bit of ambient data that affords relatively strong inferences about one or more cognitive, affective, metacognitive, and motivational states and processes" [158]. For example, consider an online textbook that allows students to highlight text online. If we knew when they highlighted the text, we could gain insight into when they were studying. If we stored what specifically was highlighted from the text, we'd be able to infer that the student thought that phrase was a particularly salient part of the text.

CS classrooms are particularly well-suited to propel trace-based analysis of self regulation forward as so much of learning occurs through digital interaction. There are many tools used throughout CS that either already track or could be leveraged to track valuable information without the biases of measures that require more of a student's attention.

Thus, to answer Research Question 2, I wrote a data and tools report assessing the software systems common to Computer Science Classrooms and determining not only the kinds of data traces available but the advantages and limitations of those traces. In writing that report, it became clear that combining data traces was not just feasible in many CS classrooms but would allow for more accurate, nuanced, and credible inferences.

This dissertation follows the outline below:

- **Chapter 2** details what self-regulated learning is and the lenses through which it is commonly viewed within the CER and LA communities.I also discuss the trends in how we measure and observe self-regulation over the last decade.
- **Chapter 3** summarizes the Delphi Process study by which we derived a list of 14 self-regulation skills that post-secondary CS educators agreed were valuable for success in CS.
- **Chapter 4** then uses that list of 14 skills as a springboard to explore what self-

regulation skills the greater CER research community has prioritized in the past. In this chapter I summarize my systematic literature review that explored this idea.

- **Chapter 5** presents a finalized list of self-regulation skills from both of these studies. Within this chapter, we also begin to answer Research Question 2 by identifying data traces for each of those skills.
- **Chapter 6** outlines the challenges still ahead for this work and discusses the existing fields of research that will help future researchers surmount those issues.
- **Chapter 7** concludes by summarizing the findings of this work, especially for multi-source approaches to data traces, and suggests some future research directions.

Statement of Attribution The large body of this work uses content from previous publications where I am the primary contributor and lead writer. In order to form a coherent narrative, some sections and subsections have been re-arranged. At time of writing, all three manuscripts are pending publication.

For full reference, the papers that have been repurposed are as follows:

1. Identifying Critical Self-Regulated Learning Skills: A Delphi Process Study [30]
2. How Can We Know When We See It? A Systematic Review of Cognitive Control Skills and Behaviors [31]
3. Taking a Step Back: Assessing the Feasibility of Using a Wider Digital Ecosystem to Study Self-Regulated Learning Skills [29]

Sentence-level changes are superficial in order to improve the overall narrative and coherence of this work.

Chapter 2

Background

2.1 What Is Self Regulation

The "Handbook of Self-Regulation of Learning and Performance" defines self-regulated learning as "the processes whereby learners personally activate and sustain cognitions, affects, and behaviors that are systematically oriented toward the attainment of their learning goals" [129]. While different models all approach the construct from different perspectives, theorists generally recognize skills like 'goal setting', 'metacognitive monitoring', and 'reflection' to all be self-regulated learning in action [164].

SRL is also closely tied with academic success [101, 163]. It is well correlated with other psychological factors of academic success like motivation, interest, and self-efficacy [86]. Perhaps because of this, self-regulation is one of these best constructs to use as a predictor of academic success [111, 164].

2.1.1 Clarifying Terms

Self Regulation vs Metacognition

While the scope of this dissertation focuses on SRL, we wish to take a moment to provide working definitions for both self regulation and metacognition. This is because literature

on self regulation and metacognition often uses these terms to convey the same concepts. Overall, "metacognition" is more associated with psychology, while "self-regulation" is more associated with education [119]. No universal or clear-cut boundary between these terms exists. Self regulation was originally considered one component of metacognition before developing into an independent construct. Thus, it is in some ways a subdomain of metacognition. Yet "metacognitive monitoring" is an aspect of many major theoretical frameworks of self regulation, making metacognition a subdomain of self regulation as well. Terms like "metacognitive self-regulation" also appear in the literature, further confusing things.

Since these words can mean different things to different people, Prather et al.and Loksa et al. [84, 119] have previously recommended that researchers create their own working definitions. Following that recommendation, we now define how we use these terms within this work.

- **Metacognition** concerns the knowledge generated from the process of 'thinking about thinking'.

- **Self Regulation** concerns the application of metacognitive knowledge to manage behavior in order to complete a task.

Ultimately, this is why I focus on Self-Regulated Learning within this research. Metacognition, according to our working definitions, is an entirely internal and unobservable process while self regulation has some behavioral antecedents.

For readers looking for a more detailed dive into the differences and uses of these terms, the review by Loksa et al. [84] is a good resource.

Other Important Vocabulary

Throughout this dissertation, I use the term self-regulation **skill** to mean something specific. A self-regulation skill refers to an internal mastery of a particular aspect of self regulation. Time management might be an example of a skill in this sense.

A self-regulation **behavior** is an observable action or evidence that indicates the presence (or absence) of a particular SRL skill. Spacing out study sessions before an exam might be a behavior that indicates strong time-management skills.

While discussed in Chapter 1, we will also take a moment to reiterate the definition of a **data trace** as defined by the Handbook of Learning Analytics. As described in Section 1 a trace is "a bit of ambient data that affords relatively strong inferences about one or more cognitive, affective, metacognitive, and motivational states and processes" [158]. For example, if a software system logged a record of when a student submitted their first draft of a homework assignment, an outside observer could make an inference regarding that student's time-management behavior. Within this work, we argue that a data trace can be more than just a single piece of data that allows for an inference but could be many pieces of data each providing some context into a student's overall behavior.

A Data trace needs to come from some piece of software. When discussing the digital education software tools that students engage to generate data traces, we use the term **data source**.

2.1.2 Theoretical Models of Self-Regulated Learning

Self regulation in the context of learning has many frameworks, each tying in other educational constructs. In their meta-analysis, Panadero et al.identify six major approaches

to self-regulation theory [101], as represented by the following papers: Zimmerman [165]; Pintrich [113]; Winne and Hadwin [157]; Efklides [36]; Hadwin, Järvelä, and Miller [52]; and Boekaerts [10].

The Cyclical Phase Model

Zimmerman's Cyclical Phase model [165], developed in 1986, remains one of the most well-recognized conceptions of self regulation with respect to learning. Stemming from his work with Bandura [8], the Cyclical Phase model divides the process into three distinct phases that a student moves through sequentially as they complete work.

The forethought phase comes before starting, when a student prepares to do a task. The task is analyzed, goals are set, and a student might develop a plan on how to approach their work. A student might also consider some aspects of motivation like their self-efficacy, expectations, interest, and personal goals for their work.

Once a student begins to work, they enter the performance phase. During this phase, students might observe themselves and make adjustments to better suit their needs. These adjustments could be strategic, like keeping track of time or seeking help when stuck, or they could be motivational, like creating an incentive for finishing.

After a student has finished their work, they progress to the self-reflection phase. Here, students look back on the work they accomplished and consider the variables that might have contributed to the quality of the work they produced. This evaluation also tends to have an emotional component as well. This self-reflection and the emotional responses of that reflection feed into the forethought phase of the next task a student does (which is why the model is called cyclical). For example, if a student performed below their own expectations, that might impact their self-efficacy considerations in their next forethought

phase.

Pintrich's Model

Pintrich's framework for SRL is commonly referenced as an influence within the CER community. While similar to Zimmerman's, this model puts an even greater emphasis on how the self-regulatory process and motivation interact [113]. Pintrich's model breaks self regulation into two dimensions: four stages of the process of self regulation and four areas that one can regulate. The four phases are:

- forethought, planning, and activation
- monitoring
- control
- reaction and reflection

At each stage, a student can regulate their cognition, motivation/affect, behavior, and context. For example, in the forethought, planning, and activation phase, a student might regulate their cognition by setting a goal. They might regulate their behavior by setting aside time to work on the task. 'Context' represents the variables surrounding the student and the task like cultural context or physical environment. While this first phase does not have a contextual dimension, the control phase can see students changing contexts, like going to the library for a quieter place to study [113]. Pintrich's model serves as the foundation for the Motivated Strategies for Learning Questionnaire (MSLQ), a well used inventory for self-regulated learning both within the CER community and the general education community [84, 126].

Winne-Hadwin Model

Most notable from the perspective of LA is Winne and Hadwin's COPES model. This is likely in no small part because Phillip Winne is a major advocate of studying self regulation using data traces and has authored chapters within two recent editions of the "Handbook of Learning Analytics" [158].

The COPES model is used within this work to derive data traces and is best described as a two-dimensional grid. There are four phases of self regulation and five variables that influence how a student engages within that phase. The four phases are:

- task definition
- goal setting and planning
- enactment
- adaptation

During the task definition phase, a student comes to understand what they are supposed to be doing. In the goal setting and planning phase, they develop their plan and establish sub-goals. Students then work through that plan during the 'enactment' phase and finally, after the task is finished, reflect during the 'adaptation' phase and make decisions for future tasks.

The 5 different dimensions identified in this model are Conditions, Operations, Products, Evaluations, and Standards (or COPES for short).

- Conditions are any variable that will influence the work.
- Operations are the ways a student processes information and behave within that phase.

- Products are any artifacts created by that behavior.
- Evaluations are the assessments a student makes of their behavior.
- Standards are the rubrics a student uses to conduct those evaluations.

These two components (phases and COPES variables) work together to describe self regulation. For example, within the task definition phase, one Condition might be how much available time a student has during the assignment's duration to complete the work. One Operation might be how they process the assignment information. One Product might be the notes they took. They might move on to studying something new when they have Evaluated that they have understood things sufficiently, where their Standards define what 'sufficiently' means in this context.

2.2 The Three Waves of Self-Regulation Measurement

The question of how to best measure SRL skills has proven difficult to answer. Panadero et al. [102] identified three major "waves" in how SRL has been measured over time.

In the first wave, self-regulation skills were first often conceptualized as static and innate traits. These traits were commonly measured through student self-reports like the MSLQ [114] or Learning and Study Strategies Inventory (LASSI) [155]. As Prather et al.put it: "Self-report measurements of cognitive control, such as the MSLQ, often measure what students think they do, rather than what they actually do" [120].

During Panadero et al.'s second wave of SRL measurement, there was a shift in the way researchers conceived of the construct. Instead of being an innate trait, self regulation

began to be seen as a series of events. Methods of study subsequently evolved with this conception, focusing more on ways of directly observing behavioral events, like Think-Aloud studies. These studies offer a better view into a what a student actually does but center their focus on how students solve individual problems outside of class. Such observations might not generalize to how a student prepares for an exam or completes an in-class assignment.

Another form of event-based measurement is to use data 'traces' to empirically track student behavior through interactions with educational software. For example, if a piece of software recorded when a student highlighted text on a page, an outside observer could reasonably make the inference that the student found that passage of text important. Instructors can use such traces to identify at-risk students and offer personalized help even in large classes. Unlike Think-Aloud studies, which observe student behavior outside of their normal classroom setting, traces capture data of students while they authentically learn. Unlike surveys, traces capture behavior rather than self-reported beliefs. This means that data traces act as an effective complement to other forms of research.

Still, as with any approach to measurement, data traces do have limitations. As they vary depending on the source those traces are coming from, those limitations are explored more in depth in Section 5.3. Broadly, data traces are limited in that they are another step divorced from the reality of a student. With other event-based measures like Think-Aloud studies, researchers are making an inference regarding a skill based on a direct observation of a student's behavior. For example, Loksa and Ko used statements like "I'm going to initialize variables first" to indicate that a student was planning during their Think Aloud study [81]. With data traces, researchers need to ensure they have two strong inferences a strong inference from data to behavior before they consider if that behavior implies a particular skill. Thus, using data traces requires two strong inferences rather than the single one needed for approaches like a Think-Aloud study.

Currently, we are in what Panadero et al.propose as the third wave of SRL research. SRL is still commonly conceived as a series of events with event-based measurements seeing wide use. However, in this third wave, the tools for measurement are also now the same tools used to improve SRL skills. Panadero cite ‘learning diaries‘ as an example of a third-wave measurement tool. To the students, the act of completing the diary helps them practice reflecting on their own learning process [102]. For researchers, the prose of the diary helps provide a view into how a student's self-regulatory processes develops over time. These diaries are therefore simultaneously providing a scaffolding for SRL and rich data for researchers. Similarly, LA dashboards have become a common artifact of third-wave research, where data traces are collected, aggregated, and shown back to students to improve metacognitive awareness [2]. Students get to immediately see trends in their behavior summed up and, by tracking visits to those dashboards, researchers can keep tabs on how frequently a student reflects on their own processes.

Chapter 3

Creating a Shortlist of Skills Part 1: A Delphi Process Study

This chapter is almost entirely text from 'Identifying Critical Self-Regulated Learning Skills: A Delphi Process Study' (under review at the Journal 'Computer Science Education'). This means the term 'we' is used. The authors on this study were (in order): Molly Domino, Bob Edmison, Stephen H.Edwards, Rifat Sabbir Mansur, Alexandra Thompson, and Clifford A.Shaffer. Thus, 'we' refers to that list of authors.

Self-Regulated Learning can encompass many qualitatively different skills a student uses before, during, and after working. Since instructors only have limited time for such training within any give course, what skills should they target? And how would they know what improved self regulation looks like? This Delphi Process Study and the systematic literature review discussed in Chapter 4 work begin answering these questions by creating a shortlist of promising skills for future targeted work.

The proposed list of skills gathered from the Delphi Process was used as an initial framework

What Came Before Previous work has largely identified self-regulation skills by either asking students to report what they find valuable [121, 136], or by pulling from more general models of self regulation [3, 28, 35, 135]. General models offer a valuable baseline, but cannot tell us anything context-specific. Asking for the perspective of students can help us identify

some ways in which self-regulation skills manifest in CS, but, as discussed in Section 2.2, self-reporting is increasingly seeing critique [108, 119]. At best, student self-reports could be biased towards what students believe they should be doing, rather than the skills they are actually using.

Additionally, that assumes students have a working knowledge SRL theory and all possible skills that fall under that umbrella. Students cannot possibly identify skills without knowing that such skills exist. Ultimately, lists of student-reported SRL skills also will miss lesser known but still critical concepts .

While preliminary research has examined what college students view as valuable SRL skills [39], we sought to broaden this discussion. Educators have a useful viewpoint to add to this conversation, as they may have a more realistic view into what students do (or should do) to succeed. Instructors also interact with many students year-over-year and so might provide insight into broader trends of student behavior.

Thus, our specific research question for the Delphi Process Study was **RQ1: According to post-secondary CS educators, what self-regulation skills are most important to learning Computer Science**?

3.1 Approach

The Delphi Process [47] is a structured process through which a consensus opinion is gathered from a panel of experts. It can be especially valuable in areas that are heavily opinion-dependent or where no conclusive evidence yet exists—both of which are the case for this work. Through a series of anonymous and iterative surveys, information is gathered, processed, and the collected ideas are presented again to the panel. This allows experts to share

their thoughts and opinions anonymously and in a way that prevents a subset of panelists from dominating the conversation [47]. Using Niederberger et al.'s taxonomy of different types of Delphi Processes [99], we concluded that a purely qualitative Delphi Process would be the best way to answer our research question as we were interested in generating a list of promising SRL skills. Niederberger et al.note that a purely qualitative Delphi Process is best for "generating and aggregating different ideas and solutions for a problem".

We constructed a three-phase Delphi process study. In Phase 1, participants were each asked to list at least five SRL skills they believed to be important to CS success, and list at least one potential behavioral indicator for each. In Phase 2, participants read over a list of skills and indicators that represented an aggregation of all Phase 1 responses, and offered feedback on what changes needed to be made. We also made an effort to better contextualize the discussion at this point, as we noticed some confusion in the Phase 2 responses. To do this, we provided a concise overview of the full skillset before asking participants to address each skill in turn. After those changes were implemented, participants again reviewed the list of skills in Phase 3, where they provided feedback and rated their level of agreement with each skill.

For all three surveys, participants were given notice that a survey was coming one week before it was sent out. Participants then had ten days to complete the survey.

Assembling a Panel– Our research question seeks to identify the SRL skills viewed by educators as most likely valuable to learning CS. Prior studies have sought to identify CS-centered SRL skills by seeking student perspectives. In this work, we sought to broaden our understanding by assembling a panel of post-secondary CS educators. To broaden that understanding, we prioritized the perspectives of experienced CS professors. While some panelists did have a research background in SRL, this was not a requirement. Our thinking was that practicing educators would be able to give the best perspective on what skills are

most valuable for success in CS, based on our working definition of SRL. A further discussion of why we focused on post-secondary educators can be found in Section 3.4.

We identified nine candidates who we thought had the appropriate experience, knowledge, and likely interest. They were solicited via direct email, and ultimately eight participants agreed to be on our panel. All participants completed all phases of the study and did so online. While Delphi processes do not have a consistent optimum size for panels, eight is within the recommended size (seven to fifteen) for a purely qualitative Delphi-Process study [98].

Our IRB deemed this work exempt from IRB review under 45 CFR 46.104(d) category 2(ii). An information sheet was provided to all participants detailing the methodology of this work, their rights, how data would be anonymized, and who they could contact with questions.

Demographic information– Four of our panel teach (at the time of writing) at public universities with between 20,000 and 40,000 students. The remaining four panel members teach at private universities where student populations range from 500 to 25,000. Our panel members currently teach at universities in Texas, North Carolina, Virginia, Delaware, and Maine. Beyond that, no demographic information was collected during the Delphi Process.

Overview of Methods Used: After the study's completion, panelists were sent an optional survey asking some demographic information. Not all participants opted to disclose this information so the following information does not fully describe our panel. Participants reported a collective 97 years of experience teaching computer science (averaged at 16.1 years of experience). Class sizes varied with some reporting teaching between 10 and 40 students in a class and others teaching between 80 to 200 students. Five participants identified as white, one as Hispanic, and one as Asian. Two participants identified as female and five as male.

Before every survey, participants were given a one-page summary of the context of the study, their rights as voluntary participants, and contacts they could reach out to with any questions or concerns. A link to download this document was also included so participants could keep records if they wanted.

Phase 1: Concept Identification In the Phase 1 survey, participants were given common definitions of SRL and what the terms "skill" and "behavior" mean in the context of this work. These definitions were identical to those provided in Section 2.1.1.

In this phase, participants were asked to list at least five self-regulation skills they considered important to success in CS. They were asked to provide a definition for what each skill meant to them and to list at least one way an outside observer could recognize proficiency or display of that skill in students. These questions were open response and had no length limit. Participants were given notice that the survey was coming one week before a link was sent. They were then given ten days to complete the survey.

Survey 1 responses were inductively coded by three members of the research team. In this methodology, codes are derived from the text itself, rather than generated beforehand externally and applied to a text. To calibrate the process, one participant's response was selected for coding by all three coders. This initial effort yielded only 49% agreement. The coding team then met to discuss their results and their reasons. After this discussion the group agreed to some informal and tentative codes and their definitions. They then independently coded the same response again and were in 100% agreement.

The remainder of participant responses were divided among the three coders. The tentative codes were applied, when possible, though any passages that did not clearly fit within this schema were highlighted. The team met once again during this process to discuss such passages and how the tentative codes should be amended to better suit the text. The fifteen

informally-defined skills created in this phase are shown in Appendix A.

Phase 2: Initial Evaluation The research team used the code book developed in Phase 1 as a preliminary list of self-regulation skills and definitions. This list was then presented to the participants for further refinement in Survey 2. For each skill, participants were first presented with the skill's name and the working definition the coding team had created. Participants were then asked what they felt should change (or not change) about the skill and its definition. This was followed by a bulleted list of potential indicators for a particular skill. These indicators were paraphrased from the behavioral indicators that participants listed in Phase 1. Participants were then asked if they felt all indicators in the list fit that skill and what (if any) indicators should be added or removed. Lastly, participants were asked if the list of skills presented fully encompassed all SRL skills that were important to CS. As with the Phase 1 survey, questions were open response and had no length limit.

At this stage, participants were providing feedback about the skills from Phase 1. Their responses either expressed that they liked one of our tentative skills and its definition or offered a critique about how to improve it. Rather than inductively code these responses as done with the first survey, the research team determined that it would be more effective to use this feedback to correct and further refine the codes generated from the Phase 1 survey.

To accomplish this, several members of the research team (some who had coded Phase 1 data and some who had not) independently read over all responses and wrote up a summary of what changes each participant wanted. These research team members then met to discuss their readings of each piece of feedback and what changes needed to be made to address their critiques. One researcher then implemented those changes. This process was repeated until research team reached 100% agreement that the list of skills and behaviors had been adjusted to address all feedback from participants.

One concerning trend the team noticed in Phase 2 responses was that some Delphi panelists had made notes to disregard feedback they had made earlier in the survey since they did not gain a full understanding of a skill until they had read over all skills on the list. In Phase 2, participants were sequentially presented with each skill on the list with no overview of all skills they would be evaluating. The research team found that responses were sometimes muddled by this presentation style. Thus, along with refining skill definitions and behavioral indicators, we also sought to find a better way to present information that would help participants get an overview of the skill set before seeing questions about individual skills. To help facilitate this, the research team created an optional supplementary document giving more detail on each skill, along with embedding Figure 3.1 into the beginning of the Phase 3 survey.

After this refinement, the research team agreed to remove one skill (*Design Approach* from the list. Participants did not feel the skill fit within our definition of self regulation. A full list of the fourteen skills and their refined definitions can be seen in Appendix A.

Survey 3: Final Rating The questions in the Phase 3 survey were similar to those on the Phase 2 survey, but the response format was different. First, participants were shown Figure 3.1 and given optional access to the more detailed definition document. Then for each skill on our refined list, participants were presented with a skill name, definition, and a set of behavioral indicators in a text box. Participants were given a five-point Likert-scale question asking them to rate how well they agreed with the statement "*[skill] is a self-regulation skill that can help Computer Science Students succeed in class*". This Likert scale consisted of five options: "Strongly Agree", "Agree", "Disagree", "Strongly Disagree" or a write-in field labeled "Other". An optional text field allowed participants to give feedback on the revised skill presented and discuss if all of the behavioral indicators were indeed indicators of the skill presented. The optional text field had no length limit.

Responses for this survey were aggregated and processed in the same way as for the Phase 2 survey. Almost no critiques were made during Phase 3, leaving little to process. We considered a consensus to have been reached if at least six out of eight of participants responded to the Likert-scale question with 'agree' or 'strongly agree', and a majority of feedback provided contained no critique. In fact, the consensus was far stronger. Ten out of fourteen skills had responses that were unanimously 'agree' or 'strongly agree', and the remaining four had seven of eight participants agree or strongly agree without critique. These last four are discussed more in Section 3.3. The fourteen skills sorted from most agreement to least agreement can be seen in Table 3.1.

3.2 A Proposed Body of Skills

At the end of Phase 3, our panel had come to a consensus on fourteen SRL skills that were valuable to success in CS, and provided some associated behaviors as indicators of skill use or proficiency. As mentioned before, between the second and third phases of the process, we realized that the Delphi participants needed to see the context of the entire list of skills before evaluating individual ones. As part of that effort to present the skills in a way that promotes better understanding, the research team began considering relationships between the various skills as a way to help introduce readers to the collection. After much consideration, we grouped the skills into five overarching categories: Planning, Executing, Monitoring, Responding, and Reflecting.

- **Planning Skills** center on how a student forms the right cognitive framework before working.

- **Executing Skills** concern how a student sets intentions and start to work.

- **Monitoring Skills** concern how a student stays aware of their current state while working.
- **Responding Skills** concern altering behavior based on that awareness of state while working.
- **Reflecting Skills** concern how a student evaluates their work after it has been completed and deciding about changes to approaching future work.

Figure 3.1 provides a full list of skills and their brief definitions. In the rest of this section, we provide a more complete description for each skill, by group.

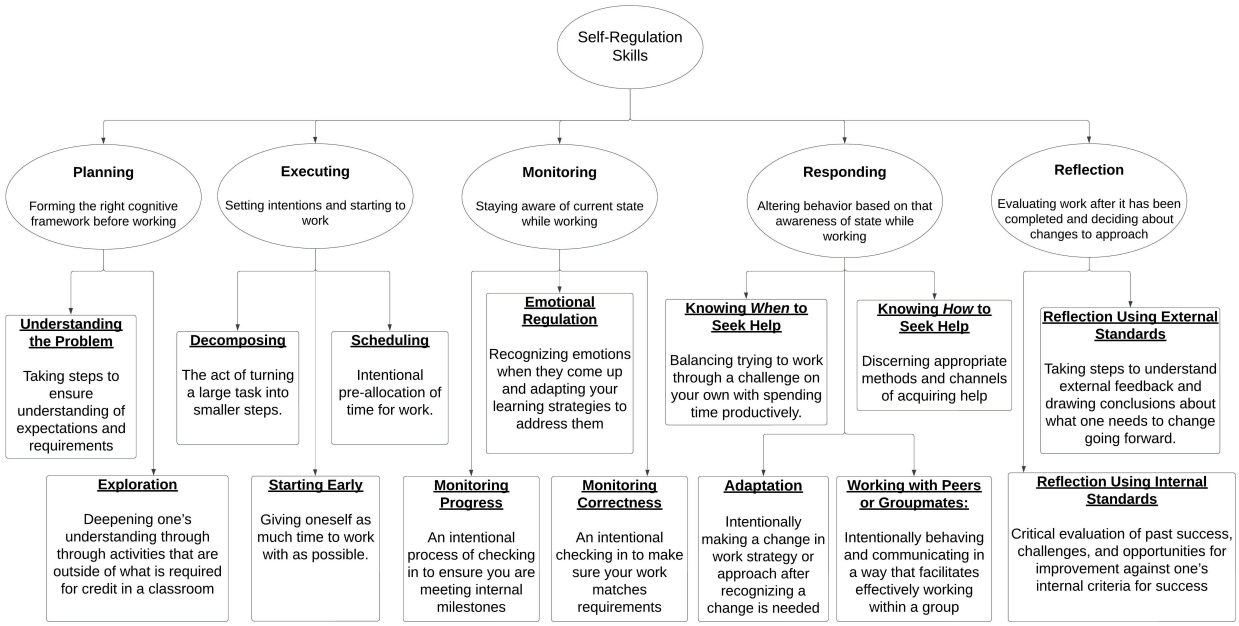

Figure 3.1: Diagram of all skills grouped by common themes

3.2.1 Planning Skills

This set of skills concerns setting intentions around how to perform a task. Theoretical analogs include the *Forethought, Planning and Activation* phase of Pintrich's mode or

the "Task Analysis" component of the *Forethought* phase in Zimmerman's Cyclical Phase model [101].

In the Winne-Hadwin model, analogs to these skills are seen in separate phases of the model. *Understanding the Problem*, with its focus on forming a correct conceptual model, is neatly summarized by the *Task Definition* phase of the Winne-Hadwin model [157] while *Exploration* falls more under the purview of *Enacting Study Tactics and Strategies*.

Understanding the Problem concerns taking time to understand and internalize the requirements of a task before jumping into solving the problem. Failing to do this often leads to incorrect conceptions of requirements which can lead to faulty solutions. While a correct conceptual model is the desired outcome, correctly understanding the question is not an aspect of self-regulation. The key to Understanding the Problem is that students take steps to form that model correctly. Panelists discussed observing this skill by seeing students re-read instructions or use other learning strategies (like note-taking) to help them understand what is being asked.

Several other studies have recognized this skill as important for success. Falkner et al.and Parham et al.both identified understanding the problem as important aspects of self regulation and metacognition in their respective works [39, 103]. In 2018, Prather et al.recognized correctly forming a conceptual model of the problem as the "most glaring inconsistency'' between successful and unsuccessful students in their Think-Aloud study [117] and later found promising results creating an intervention to address this [118].

Exploration describes the process of deepening one's understanding through activities that are outside of what is required for credit in a classroom. This skill is used when a student is furthering their understanding or assessing what tools are best for the situation. Tinkering, or making changes to code to experiment and learn (rather than align with assignment

requirements), was frequently discussed by our panel as a potential indicator.

While there is some discussion in the CER community, Exploration could be a good target for future focused study. In their qualitative analyses of student-reported skills, both Prather et al.and Falkner et al.identified practicing writing code as an important yet infrequently discussed behavior [39, 121]. Independently investigating questions external to the class has also been cited by researchers as a potential indicator [39, 109].

3.2.2 Execution Skills

These skills concern the process of establishing a plan and starting to work. Early on, the term 'planning' was inconsistently used by our panel to mean intention setting (like those we've categorized as 'Planning skills'), time management, or strategizing. This seems to be mirrored in most models of SRL used today, where "planning" serves as an umbrella term for these ideas [101]. Indeed, Zimmerman's Cyclical Phase Model includes aspects of decomposition and time management as part of the forethought phase under the term "goal-setting and planning" [165].

In order to ensure consistent vocabulary among participants, the research team has broken out what appear to be the distinct skills one uses to strategize how and when they will work.

Decomposing was brought up by six of our eight panelists in Phase 1. This skill is used when it is important to break the task into smaller, more tractable steps to help them prioritize and set goals. Students must also understand how these goals interrelate and how they contribute to the overall needs of the task. One way to see Decomposition in action is to observe some of the supportive artifacts created like project boards or burn-down charts. Inline comments on code can provide a view into the sub-goals a student recognized in a task. In courses where automated assessment tools are used, it may be possible to identify

code submissions where work is done on a single feature at a time.

Decomposing is frequently highlighted as an important self-regulation skill for CS students [3, 39, 103, 145]. In their Think-Aloud study, Loksa and Ko have found that the frequency of statements concerning goal-setting or revisiting previously stated goals was significantly related to fewer errors in code [81]. However, at time of writing, few studies have shown a statistically significant link between this skill and broader success metrics.

Scheduling is making an intentional decision to set aside time for a particular activity. It requires a student to assess how long they expect a piece of work to take and to know how long they can successfully work in a given session. In Phase 1, five of eight panelists discussed the idea of "keeping track of time" or "distributing tasks over a schedule". When asked how to observe this skill, panelists discussed ways of knowing when a student is doing their work. For example, tracking when submissions are made to an auto-grading software or a Learning Management System would help professors get a view into how students were using their time. Additionally, procrastination is considered an inability to Schedule since, while it can occur for many reasons, ultimately results in a difficulty with allocating time to work.

In other research, Scheduling sometimes is discussed in terms of spacing study sessions out versus cramming [3, 18, 121]. This seems like one of the most tangible ways to see Scheduling in action since spacing out study sessions inherently seems to imply that a student is making decisions about how to manage their time. Procrastination is a much more well-studied aspect of this skill [3, 24, 35, 39, 78, 90, 130, 162].

Starting Early is a skill that allows students to optimize their time. Participants in Phase 1 initially included this as an aspect of time management, but it became clear through discussion that Starting Early applies to all tasks including ones that do not require an explicit

Scheduling or Monitoring component for success. One can observe Staring Early when students first begin to interact with resources (like when a student first submits to source control or first seeks help). These behaviors imply that the student has started to work.

We only highlight the outcomes of two studies here but, Starting Early is often called out as an important SRL skill for computing [3, 26, 74, 117]. Arakawa et al.use "not starting [a project] before 1/2 of the allowed time has elapsed" as an indicator that a student is struggling to self-regulate [3]. Prather et al.found a weak but significant correlation between how early a student started a homework assignment and the course exam grades [117].

3.2.3 Monitoring Skills

The performance phase of Zimmerman's Cyclical Phase model is decomposed into two categories: "self-observation" and "self-control" [164]. These skills focus on different aspects of a student's mental state that they need to be mindful of and are therefore all aspects of self-observation. Decisions made in response to this awareness, what Zimmerman calls "self-control", are covered in Section 3.2.4.

Monitoring Progress is about recognizing one's own pace of work and potentially noticing when one is stuck or moving slower than anticipated. As this awareness is highly cognitive, identifying when a student is aware of their progress through a task can be difficult. One straightforward indicator discussed in the literature was simply to assess if students can articulate their awareness. This could be as simple as asking students to report on how close to finished they believe they are with an assignment.

The name of this skill was adapted from Loksa and Ko's list of self-regulation skills for problem-solving. They note "programmers who explicitly monitor their progress toward solving a problem are more successful (e.g., is a sub-goal complete? Is the code sufficiently

tested?)" [81]. Prather et al.similarly noted a "poor sense of location" was a major impediment for success in their Think-Aloud study as students would be unwilling to evaluate their overall approach because they incorrectly believed themselves to be nearly finished [117].

Monitoring Correctness is the ability to question and validate that one's work matches the requirements of the task (such as the requirements laid out for an assignment). When panel participants discussed this idea, it was usually in the same sentence as Understanding the Problem. However, in coding this theme, it became clear that this was a skill students used not just when they started their work but throughout the process of working. Some participants also wanted to further distinguish Monitoring Progress from Monitoring Correctness, since monitoring and updating a timeline can be done independently from taking steps to be confident that work done is correct. Educators can see this skill in action when students confirm that code is working as expected after making a change.

Loksa and Ko observed what they called "Comprehension Monitoring" when students evaluate their implementation of code "usually by testing and debugging" [81]. Automated Assessment Tools (AATs) offer a different way to find evidence that a student is validating as they offer feedback about the correctness of the code submitted. Arakawa et al.used "fail[ing] to pass the same test case in three sequential commits & pushes to GitHub'' [3] as an indicator that a student might not be engaging with AAT feedback effectively and might be struggling to self-regulate.

Emotional Regulation requires an awareness of emotion and choosing how to act in light of those feelings.

This skill saw a great deal of refinement over the course of our study. In Phase 1, managing frustration, patience, and resilience towards failure were touched upon by two participants, and the initial code created during Phase 1 analysis focused on solely working through these

negative emotions. In Phase 2, we received strong feedback that this code needed to be broadened beyond just those feelings and focus more on mindfulness rather than "pushing through" emotion. Difficult emotions are likely the easiest ones for observers to see, and indeed, when discussing indicators for this skill, participants overwhelmingly discussed ways they would see a student struggle to stay mindful of their feelings. For example, one indicator might be observing when students quickly attribute failure to causes outside of themselves like the compiler, language, or AAT. However, this is only one aspect of a much greater skill. Awareness of all emotions is important. Regulating excitement, for example, can help students resist the urge to jump into programming before reading a problem prompt fully.

Emotional Regulation was initially brought up by two participants in the Phase 1 survey, but was unanimously agreed to be important in the Phase 3 survey. In their review of e-learning tools that support SRL, Falkner et al. [45] noted something similar. Emotional Regulation was a consistent theme among self-regulation supporting software, yet it was absent from the theoretical framework they were using.

3.2.4 Response Skills

Going hand-in-hand with Monitoring Skills, Response Skills are focused on what a student does once they are aware that they need to make a change. These skills constitute the "self-control" aspect of Zimmerman's performance stage [164].

Working with Peers or Group Mates concerns the ways a student intentionally works and behaves differently on a task when interacting with peers than if doing the task alone. This skill requires an awareness of the ways in which the individual's default process of working may not be successful for the group. While indirect, one tangible way to see this skill is to compare submissions from group assignments versus the member's individual assignments

and look for qualitative differences.

Hadwin et al.define 'co-regulation' as the "dynamic metacognitive process through which self-regulation and shared regulation...are transitionally and flexibly supported and thwarted'' [51]. Prather et al.have conducted some preliminary investigation into what co-regulation looks like in a CS classroom [121], but this topic needs more exploration.

Knowing When to Seek Help is about balancing trying to work through a challenge on one's own with spending time productively. When it is possible, one indicator for this skill may be to observe students using a formalized system for timing help-seeking appropriately. Observing a student setting a timer to seek help after an hour if needed is an example. Panel participants listed some indicators that could help identify when students are struggling to use this skill effectively, such as when a student does not seek help until after a due date.

CER studies are still working to identify the full depth of nuance in this skill and there has been little work relating it to academic success. Marwan et al.identified several forms of unproductive help-seeking broadly categorized as help over-use (or abuse) or help avoidance [92]. Doebling and Kazerouni have also observed that help-seeking is closely tied to novice student's sense of identity and they might under-use available help in order to support that sense of belonging [28]. Like everything related to this skill, this specific issue of sense of belonging and help-seeking deserves further exploration since research shows that individuals from traditionally underrepresented groups in CS have a lower sense of belonging [72].

Not all help-seeking approaches are equally valuable. The skill of **Knowing How to Seek Help** is used when a student discerns appropriate methods and channels of acquiring help. *Methods* refers to how a student forms the questions they ask. *Channels* refers to the resources a student chooses to use when seeking help. While this skill is called *Knowing* How to Seek Help, this skill requires more than an application of knowledge. As one participant put it,

"There are lots of social and logistical barriers to seeking help as well, so part of it is an awareness of these barriers and the ability to overcome them''. This perspective is consistent with other literature on the field as well [28].

Discussions of indicators for this skill were centered on observing how students interact with course resources like office hours, discussion boards, and Learning Management Systems (LMS). Conversely, panelists saw asking generalized questions like "why is it broken?" as an indicator that a student was still working to develop this skill.

Doebling and Kazerouni observed a trend that students tend to exhaust resources that are easily accessible but lower utility (like online sources) in preference to those that are less accessible but higher utility (like an instructor) [28]. Ren et al.created a formalized framework for students to ask questions in office hours [125]. This "Design Recipe" served as both intervention and measurement tool. Students recognized it as a valuable resource in observing office hour interactions and results demonstrated that it helped scaffold thoughtful question asking.

Adaptation is about how a student chooses to make a change once they recognize that there is a problem with their approach. To use this skill, students must consider the tools at their disposal and then make a choice about which tool is most appropriate for the situation. This skill was discussed by many participants in Phase 1 when discussing students intentionally using problem-solving strategies to help them complete an assignment, but Adaptation could look like any change in behavior to an outside observer.

This skill was separated from the two help-seeking skills in this category, as making the decision to seek help was qualitatively different from making that decision at the appropriate time and in the appropriate way.

In the literature, help-seeking and searching for analogous problems are commonly discussed

methods of Adaptation [39, 78]. The key for recognizing adaptation in action is to recognize when a student is stuck. Once they become stuck, observing what strategies they take can indicate whether they are displaying the Adaptation skill.

3.2.5 Reflection Skills

Whereas Adaptation is used to make fine-grained changes while working, Reflection Skills are focused on appraisals of a finished task and the process of considering larger-scale adjustments for future work.

One could reflect after completing a whole assignment, but stopping to reflect after completing a sub-goal is also possible. Pintrich, Zimmerman, and Winne and Hadwin's models of SRL all include a similar appraisal phase after a student has finished a task [101].

The two skills in this section focus on different sets of Standards as defined in the Winne-Hadwin COPES model. They define Standards as the criteria against which the output of a task is evaluated [156]. In Reflection Using Internal Standards, we consider a student's personal criteria for success and Reflection Using External Standards considers the criteria for success dictated by the official rubrics. Research on these skills in CS classrooms is present but sparse [45, 135].

Reflection Using Internal Standards occurs when a student reflects on the outcome of a task against their own internal expectations. For this skill, Standards refers to expectations a students has of themselves. A passing grade could be a standard of quality, as could a sense of demonstrating understanding, but what matters is that students evaluate the quality of work they have produced against their own criteria for success. Without knowledge of what a student's standards are, it is difficult to say if changes in behavior are done in order to align work with their expectations. While indirect, panelists discussed hearing students articulate

their attributions as to why they received the grade they did as a method of observing this in action.

In CER, this skill is often studied by examining student responses to exams and, at a larger scale, considering ways to improve the single-try exam model to better scaffold this skill [58]. Stephenson et al.created an intervention where students had to write a reflection on their study habits and what changes they wanted to make for the next exam in an effort to make students to engage in this form of reflection. However, they found this kind of scaffolding did not change long-term behavior [144].

Reflection Using External Standards is about processing and internalizing the feedback you receive to improve according to course standards, rather than a student's personal standards. In some situations, internal and external standards might align perfectly, but that is not universal. Even when a student seeks to meet the standards of the course, internalizing external feedback may not always come naturally. One of the indicators panelists most discussed for this skill was observing when a student makes demonstrable changes to their next assignments based on what they got wrong previously. Another indicator discussed was seeing students seek out course staff to clarify feedback on a graded assignment in order to learn.

In their Think-Aloud study, Arakawa et al.noted that a "lack of reflection on feedback'' was a significant type of SRL struggle observed among their students when engaging with automated feedback tools [3]. Prather et al.also observed that engaging with feedback from their automated assessment platform was a major difference between successful and unsuccessful students [117].

3.3 Discussion

SRL skills can help students succeed, but even college students greatly benefit from instruction [83]. However, it can be difficult to decide what aspects of self regulation are most important within the context of CS. Thus, to help identify likely targets for future instruction and investigation, we sought to answer this question: **According to post-secondary CS educators, which self-regulation skills are most important to learning Computer Science?**

At the end of our three-phase qualitative Delphi Process study, our panel discussed and reached consensus on fourteen self-regulation skills that they agreed were important for CS students. These skills are all analogous to at least one major theoretical model of SRL, often more than one. All of these skills have been discussed as valuable for CS students in other research as well. For many of these skills, there is already evidence that student proficiency is a valuable predictor of academic success.

However, there is still much more work to be done. While some skills found by our panel have a healthy body of associated research, we are only just beginning to understand others. For example, Starting Early and Scheduling are widely discussed as important within the CER community [3, 18, 118, 121, 125]. However, Emotional Regulation was not frequently surfaced in Phase 1, but participants widely agreed in subsequent surveys that it was important and under-discussed, which is consistent with other assessments of the field [45].

It is our hope that the results of this study will lead to further exploration of these less-examined areas of SRL. Calling attention to these skills might encourage educators to create effective lessons on the self-regulation skills they see are most needed in their classes.

3.3.1 Identifying the Most Promising Targets

In this section we suggest which skills might be the most promising for future research.

Skills with Unanimous Agreement: In Phase 3, participants reviewed each skill and were then asked to respond to a 5-point Likert-scale question. This question asked participants to rate their agreement with the statement: "[skill] is a self-regulation skill that can help Computer Science Students succeed in class". Participants could respond with "Strongly Agree", "Agree", "Disagree", "Strongly Disagree", or write their own response if their answer was more complex. Table 3.1 aggregates the responses to that question for each skill. While the majority of skills received a mixture of "Strongly Agree" and "Agree" responses, looking at these ratings, three skills stand out.

All participants strongly agreed Understanding the Problem, Knowing How to Seek Help, and Knowing When to Seek Help were important SRL skills necessary for success in CS.

Category	Skill Name	Strong Agree Votes	Agree Votes	Disagree Votes	Strongly Disagree Votes	Other Votes
Planning	Understanding the Problem	8	0	0	0	0
Response	Knowing When to Seek Help	8	0	0	0	0
Response	Knowing How to Seek Help	8	0	0	0	0
Execution	Scheduling	7	1	0	0	0
Monitoring	Monitoring Correctness	7	1	0	0	0
Response	Adaptation	6	2	0	0	0
Execution	Decomposing	5	3	0	0	0
Monitoring	Emotional Regulation	5	3	0	0	0
Monitoring	Monitoring Progress	4	4	0	0	0
Reflection	Reflection Using External Standards	4	4	0	0	0
Planning	Exploration	4	3	1	0	0
Reflection	Reflection Using Internal Standards	3	4	1	0	0
Response	Working with Peers or Group Mates	2	5	0	0	1
Execution	Starting Early	4	3	0	1	0

Table 3.1: All Fourteen Skills from Most Agreement to Least

Previously Unidentified Skills: In order to evaluate how this instructor-created list compares to previous lists of SRL skills, we sought to compare and contrast the results of this study to previous works. Table 3.2 details places where the results of this work overlap with previous lists of SRL skills deemed valuable for CS. When creating this table, we sought to relate our skill names to the names used in prior research and their definitions to identify the closest matches. For example, while Loksa and Ko include 'Reflection on Cognition' in their list of SRL skills, they define it by saying "learners should make judgments about the qualities and limitations of their memory and reasoning (e.g., am I forgetting something? Am I making any assumptions?)" [81]. This definition emphasizes a student being aware of their mental state while working, rather than an appraisal after working. Thus, even though 'reflection on cognition' has the word 'reflection' in its name, we determined it to be

analogous to Monitoring Correctness.

Monitoring Correctness and Decomposing saw strong agreement by our panelists and are elements of all three previous CER lists of SRL skills [39, 81, 121], implying these skills are highly valued by both professors and their students.

Conversely, there are five skills that have, thus far, not been mentioned in previous lists of CER-specific SRL skills: Starting Early, Working with Peers and Group Mates, Knowing When to Seek Help, Adaptation, and Reflection Using External Standards. We note that both Falkner et al.and Prather et al.generated their lists from qualitatively analyzing student responses about what SRL skills they need. These five areas then may represent SRL skills that students under-value and thus be good targets for future investigation.

While all fourteen skills identified here are promising areas for future study, Knowing When To Seek Help was seen as highly valuable by educators and may be underrated by students. As described in Section 3.2.4, this is also an area of SRL that researchers are still just beginning to explore within CS. Thus, **Knowing When To Seek Help** may be an especially valuable SRL skill for future research to target.

For educators looking for more of a known quantity, Monitoring Correctness was recognized as important by our panel with seven ratings of "Strongly Agree" and one "Agree", and appears in all three previous SRL lists from the CER literature. While limited in scope, there is some consistent evidence that lower-performing students overestimate their correctness on exams, which implies they struggle more with this skill [24, 73, 143]. Thus, with this established link to success and wide agreement from the research community, **scaffolding of Monitoring Correctness may be an easy way to help improve SRL skills for struggling students.**

Delphi Process Skill		Falkner et al. [39]	Prather et al. [121]	Loksa and Ko [81]
Understanding the Problem	Taking steps to ensure understanding of expectations and requirements	Understand Question		Comprehension Monitoring
Exploration	Deepening one's understanding through activities that are outside of what is required for credit in a classroom	Practice Writing Code Access Resources	Coding Practice	
Decomposing	The act of turning a large task into smaller steps	Decompose Problem	Goal-setting and Planning	Planning
Scheduling	Intentional pre-allocation of time for work	Allocate Time for Prototyping Time estimation		
Starting Early	Giving oneself as much time to work with as possible			
Monitoring Progress	A regular process of checking in to ensure you are meeting internal milestones		Time-Management	Process Monitoring
Monitoring Correctness	Consistently checking in to make sure your work matches requirements	Comprehensive test cases	Self-Explanation	Comprehension Monitoring Reflection on Cognition Self-Explanation
Emotional Regulation	Recognizing emotions when they come up and adapting your learning strategies to address them	Avoid Sources of Anxiety		
Working with Peers/Group Mates	Intentionally behaving and communicating in a way that facilitates effective work within a group			
Knowing When to Seek Help	Balancing trying to work through a challenge on your own with spending time productively			
Knowing How to Seek Help	Discerning appropriate methods and channels of acquiring help	Ask Friends Talk to friends or lecturers	Help-Seeking - Social	
Adaptation	Intentionally trying another strategy after recognizing you are stuck			
Reflection Using Internal Standards	Critical evaluation of past success, challenges, and opportunities for improvement against one's internal criteria for success	Reflection and Changing Strategies Self-Assessment		
Reflection Using External Standards	Taking steps to understand external feedback and drawing conclusions about what one needs to change going forward			

Table 3.2: Overlap With Previous Lists of SRL Skills

3.3.2 Lingering Disagreements

We want to emphasize that ten of fourteen skills had unanimous support, and the remaining four had a single vote that was not 'agree' or 'strongly agree'. Put another way, the set of fourteen skills were supported by one-hundred and eight positive votes and only four non-positive votes (3%), an astounding level of agreement. Nonetheless, it is worthwhile to closely examine the four instances of disagreement as they highlight some of the nuances of these skills.

Exploration: One participant expressed doubt that this was self-regulatory at all and argued it was a byproduct of intrinsically motivated students. On the other hand, participants often suggested methods of inspiring this behavior in students. After much discussion among the research team, we find that while students who have intrinsic motivation are at an advantage, it is not a requirement for Exploration. Ultimately, this skill is about developing a mindset that aids in success. For example, reading over an external API beyond what is

needed immediately to solve a bug in code may be an efficient way for a student to gain longer-term learning that will help to achieve the good grade they seek, not simply because it is enjoyable.

Starting Early: In Survey 3, one participant strongly disagreed that this was a skill distinct from Scheduling, and suggested that the two should be merged into a "Time-Management'' skill. The research team sought to draw a distinction between setting aside time and starting to work in a way that optimized available time. However, in the Phase 2 survey, a similar critique was brought up by a different participant (who offered no critique in Phase 3). It is possible further evidence will eventually support combining Scheduling and Starting Early.

Working with Peers/Group Mates: One participant did respond 'Other' to our Likert-scale question on the grounds that this skill was co-regulation and thus distinct from *self* regulation. To reiterate Hadwin et al.'s definition, co-regulation is the "dynamic metacognitive process through which self-regulation and shared regulation...are transitionally and flexibly supported and thwarted'' [51]. From this definition, co-regulation is about how a *group* influences individual *members* to be successful. This is an important distinction to highlight. When coding this skill during Phase 1, the research team saw this as an individual's choice to change behavior rather than the group's behavior. However, it is not entirely clear who is regulating whom in group-work. It can be difficult to discern between a student individually choosing to change their workflow, and a student who felt tacit group pressure to change. Thus, this skill may not entirely be within the bounds of SRL, but there appears to be universal agreement that it is an important skill for CS students, and the research team opted to keep it.

Reflection Using Internal Standards: One participant felt this was a good skill for CS students to have, but did not believe it was required for success in the classroom. The purpose of this study was to identify a set of skills that would likely be valuable targets for

instruction and intervention, that lead to longer-term success. Thus, we find that while this participant disagreed that the skill was critical in the short term, they still acknowledged its value. It remains to be seen whether this skill stands up as something whose explicit practice will lead to improved success in the same way as it seems certain that other skills on the list will do.

3.4 Threats to Validity

With any Delphi Process study, one threat to validity is the selection of the panelists. We prioritized the perspectives of experts in CS education over experts in SRL per se. Some participants were quite familiar with SRL as an educational construct, and one is a noted expert. Other participants needed to be brought up to speed on what SRL was and was not. In providing that structure, we might have biased our participant group towards our own understanding of what does and does not constitute self-regulation.

There were times in Survey 2 when the main feedback about a skill seemed to be confusion on the part of the panelist. There were instances of five different participants giving completely (and sometimes contradictory) different feedback about how a skill should change. In these situations, the research team needed to make judgement calls about whether the problem was in our intended definition, or in how it was presented to the panel. We opted for the conservative approach of changing how skills were presented in the Phase 3 survey more so than addressing every piece of feedback from every participant. But the changes to the presentation were extensive, as the team put a great deal of effort into reworking the presentation. This included a major effort to provide an overview of all the skills through succinct definitions for each skill and with grouping similar skills into categories. The result was Figure 3.1, which was intended to provide panelists with an overall understanding of the

skill set and their relationships **before** they were asked to address each skill in turn. The panel appeared to find this an effective aid to understanding.

Forming a consensus over a series of anonymous surveys presents unique challenges. One major issue is that not all terms have universally agreed definitions. The research team found that participants had different interpretations for what constituted terms like "planning" or "help-seeking". When participants came up with qualitatively different definitions for terms, the research team endeavored to change the language on the next survey to make it specific enough that all participants could use a word to mean the same thing. However, there is no way to guarantee this worked as intended or that extra specificity was added everywhere it was needed.

While eight panelists were sufficient for our qualitative Delphi-Process Study, eight CS professors cannot possibly represent the perspectives of CS professors as a whole, or the CER community as a whole. We want to take some time to discuss our choices in focusing in on such a specific population and some of the ways that limits our study. We wanted to focus on educators, rather than other stakeholders in a student's education like professional software developers, because we wanted a panel of people who are accustomed to watching CS students and would have the experience to readily identify successful learning behaviors from a large pool of students. We assumed that students in post-secondary environments are more likely to be living on their own or transitioning to a less structured form of schooling for the first time and thus would have need of different SRL skills (and have different skills developed) from K-12 students. Thus, we limited our discussion to post-secondary CS professors as that would mean all panelists interacted with students that predominantly had the same needs. Furthermore, previous SRL skill lists from within CS primarily study the experiences of college CS students, making it easier to compare and contrast to previous work. Additionally, the panelists, at time of writing, predominantly teach at large US universities

with a population of more than 20,000 students. This is, without doubt, a limitation of this panel and we acknowledge that broadening this conversation to include a wider array of participants is a necessary first step for any expansion on this work.

Since the main product of this research is a list of skills, the list could simply be wrong, either by including SRL skills that turn not to be important to success or missing ones that are. As documented in detail elsewhere in this paper, the listed skills all appear to have a history of being recognized in prior research. This does not itself mean that they are important to success. One purpose for publicizing this list is to encourage additional validating research. Missing relevant skills on the list is a valid concern, especially as the initial phase of the Delphi process did not seed the panelists with an initial list. Phase 1 merely asked panelists to list a limited number of skills that they considered important. So, it seems that some could easily be overlooked by this approach. We do note that from our study of the relevant literature, we dd not find important gaps in the list.

There are still so many unknowns when it comes to the study of SRL within CS. The ultimate goal of this work is to create a tangible set of targets for future researchers to examine and educators to teach. We hope that this list can be used to help provide some likely avenues for such future work, regardless of the outcome.

Chapter 4

Creating a Shortlist of Skills Part 2: A Review of the Literature

This chapter is almost entirely text from 'How Can We Know When We See It? A Systematic Review of Cognitive Control Skills and Behaviors' (under review at the journal 'Transactions on Computing Education'). This means the term 'we' is used. The authors on this study were (in order): Molly Domino, Alexandra Thompson, Alexander Hicks, Alan Jamieson, Khushi Parajuli, Bob Edmison, Stephen H.Edwards, and Clifford A.Shaffer. Thus, 'we' refers to that list of authors.

While the Delphi Process gave us a valuable view into what SRL skills CS professors valued, our work only represents the consensus of eight professors. In order to gain a broader sense of the field, we then conducted a systematic literature review of what has been studied in post-secondary CS classes. To determine if any of these skills should be prioritized for instruction, we also examined what (if any) connection researchers had drawn between these skills and academic success.

For this work, we expanded our focus from self regulation to cognitive control, which is an umbrella term for the study of self regulation and metacognition. This expansion is due to the fact that there is considerable overlap in how both terms are used in different fields and there are no universally agreed definitions. Thus, in order to cast the widest possible net, we examined what self-regulatory and metacognitive skills have been examined in post-

secondary CS classes.

Our research questions for the systematic review were as follows:

- **RQ1:** What cognitive-control skills have been examined in post-secondary computer science education research, and what behaviors have been used to indicate skills?
- **RQ2:** Which cognitive-control skills (or associated behaviors) have a demonstrated relationship with post-secondary student success in computer science education?

What Came Before: Other Reviews of SRL within CS At time of writing, no CS-specific models of cognitive control exist. The closest thing would be Loksa et al.'s model describing the self-regulatory steps that a student must engage with when problem solving [82]. These steps have been used to create Loksa and Ko's five types of self regulation that support problem solving when programming [81]. While these forms of self regulation did influence the skills identified, we seek to broaden our research beyond what students do when solving programming problems, and examine all skills a student would use when learning computer science.

Within CS, researchers have largely drawn from general education models to inform their understanding. Loksa et al.'s recent literature review provides an in-depth examination of how different works within CS education have effectively leveraged the theories of Zimmerman, Pintrich, and Flavell (among others) to better understand cognitive control when learning to program [84].

A taxonomy of skills developed by Zimmerman and Martinez-Ponz has been used by two literature reviews to map the skills they observed to a general education model. A systematic review conducted by Silva et al., explored how interventions sought to improve self-regulation behaviors [135]. With an intervention addressing each phase seen in over half of their cor-

pus, Zimmerman's three phases of *forethought*, *performance*, and *self-reflection* were all well represented. However, they noted that several strategies included in the Zimmerman and Martinez-Ponz taxonomy were not present in their review, including "environment structuring", "rehearsing and memorizing", and "reviewing texts". Silva et al.also observed that while self regulation was associated with success overall, efficacy was "not consistent across studies" [135]. Garcia et al.'s review of e-learning tools for CS used the same taxonomy to identify areas where these tools were supporting self-regulation skills [45]. Ultimately, they noted "self-evaluation" and "goal setting and planning" were the most well-supported skills while "environmental structuring" and "seeking social assistance" were least supported. Additionally, while not part of the original taxonomy used for categorization, the authors noted Emotional Regulation was an emergent theme of e-learning tools that supported self regulation. While their work is similar, Silva et al.focused on studies detailing interventions targeting SRL, while Garcia et al.focused on software that supports SRL in some way.

In the systematic literature review discussed in this section, we expand our scope to research that examines cognitive control in any context, rather than focusing on only interventions as previous authors do. We also do not start from the set of skills identified in the Zimmerman and Martinez-Ponz taxonomy as it was created from qualitative interviews with high school students in 1986. This is considerably different from the population of students we examine, which is modern college students learning to program. Modern students also have access to different resources than those available in 1986, most notably the internet.

4.1 Approach

To answer our research questions, we broadly surveyed the literature examining cognitive-control skills (and their indicators) in post-secondary CS education. We follow Plüddemann

et al.'s methods for a rapid review [115], which adapts a systematic review approach to a more limited reviewer pool while still seeking to mitigate as much bias as possible. This means while every effort was made to follow the PRISMA 2020 standards for systematic reviews, one researcher was available to select studies, and extract data. In an attempt to limit potential biasing effects of the rapid-review process, another member of the research team verified a random sampling of papers after each filtering step. Additionally, the authors sought to establish a set of clear rules for filtering at each phase to minimize subjective judgments during study selection and data extraction.

To answer **RQ1**, the research team read through each paper in the corpus and made note of passages where skills or behaviors were explicitly recognized as either self regulation or metacognition. These passages were then qualitatively coded to indicate references to cognitive control skills, behavioral indicators, and discussions of difficulties. To answer **RQ2**, the research team also noted what (if any) relationship to academic success a given paper claimed for any skill or behavior.

Figure 4.1 provides an overview of how we selected studies for our corpus.

4.1.1 Eligibility Criteria

In order for a paper to be eligible for inclusion in our corpus, it had to have all of the following characteristics.

1. The work must either be in English or have an English translation.
2. Cognitive control had to be central to a research question or analysis of results.
3. The work had to be a research paper.Case studies, experience reports, abstracts, and so on are valuable, but we are focused on how cognitive control has been studied in

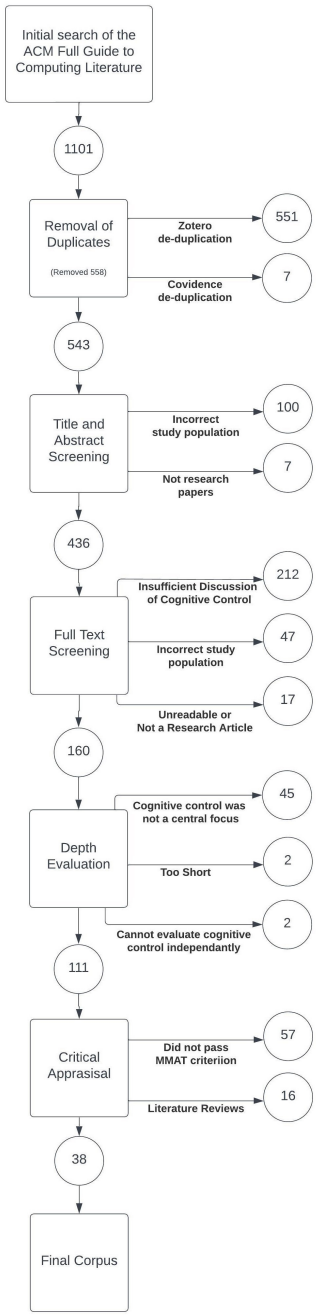

Figure 4.1: Study Selection Diagram

experimental (or quasi-experimental) settings.

4. The study's population must be focused on learning CS at a post-secondary level. We believe cognitive control likely looks different for students experiencing the less-structured nature of post-secondary schooling than it does for students in other environments.

5. The work must identify specific skills (or behaviors) which represent SRL.

6. Skills (or behaviors) must be measured or observed in the paper itself, not just within cited works. In particular, this means that we did not consider literature review papers.

The numbers from this list will be used as identifiers throughout the discussion of our methods.

4.1.2 Initial Search

To the best of our ability, we sought to generate a corpus broad enough to accurately capture the state of study on cognitive control in post-secondary CS education. Therefore, we wanted to ensure we were including all major publications of CS education research. To create a preliminary set of papers, we created and refined a search string to use on the ACM Full Guide to Computing Literature online library. This database is described as the "most comprehensive bibliographic database focused exclusively on the field of computing" [42] and includes both journal and conference publications from the ACM and IEEE presses and other outside journals. A detailed list of all included publication sources can be seen at the ACM-DL website [42]. The comprehensiveness of this resource allowed our team to use one search query, rather than face the complication of replicating the query on multiple incompatible databases.

Our search query is shown in Figure 4.2. This query was adapted from those used in two other recent systematic literature reviews on cognitive control in computer science education [84, 119]. Using the ACM Digital Library's updated search feature, this string searched the full text of all studies by default—title, abstract, text, and citations—for matches to these terms. All papers returned this way had to have some variant of the word "programming" and some variant of either "metacognition" or "self regulation". We limited our search to only include papers the ACM-DL categorized as research articles to exclude several kinds of publications that could not examine cognitive control in sufficient depth. These included proposals, posters, and published abstracts.

This search string includes both wildcard searching (such as "self-reg*") and explicit variants of the word. This is because the ACM digital library search tool does not support searching for variations of keywords automatically. The guide for using this search tool suggests using wildcards (like self-reg*) to search for variations of a word. However, after some experimentation, the research team found that only using explicit variants of a word (like "metacognition" vs "metacognitive") produced results not returned when only using wildcard searching. This effect also happened in reverse: using the wildcard self-reg* produced results not returned when searching for explicit variants of the word. Thus, both styles of search terms were included for "metacognition" and "self regulation" in order to help identify all relevant papers.

We also found that the term "self regulation" was sometimes used outside of the study of education entirely. Studies where a computer would be regulating its own decisions (self regulation) rather than a governing body (government regulation) composed a significant percentage of this initial search. We found adding "learning" to the search query helped return works relating to learning programming.

Our search was conducted on August 17th, 2022. No papers published after that date are

("program" OR "programs" OR "programmers" OR "programming" or program*) AND ("metacognition" OR "metacognitive" OR "metacognitively" OR metacog* OR "self-regulation" OR "self-regulated" OR "self-regulatory" OR "self-regulating" OR self-reg*) AND ("computer science education") AND "learning"

Figure 4.2: Updated search term used on the ACM Full Guide to Computing Literature

included in this work. There was no limitation on how old a publication could be.

4.1.3 Study Selection Process

Removing Duplicate Papers

The initial search yielded 1,101 works. Citations were bulk downloaded in groups of 50 and imported into the citation manager Zotero. This software identified many papers that had at least one (sometimes two or three) duplicates. Using Zotero's de-duplication tool, we manually confirmed each potential duplicate from Zotero and removed 551 studies from our pool.

The remaining 550 works were imported into the literature review tool Covidence. Before screening began, Covidence identified seven additional papers that had more than one entry. After manually confirming these were duplicates, they were also removed.

Title and Abstract Screening

Our corpus at this phase consisted of 543 works. In the next step, the reviewer examined only the titles and abstracts of each work to quickly remove clearly irrelevant papers.

The following criteria were used to exclude papers at this stage:

- The abstract explicitly notes that self regulation or metacognition are not a primary area of focus, violating eligibility criterion 2.
- The abstract explicitly notes that the students in question were not studying CS in a post-secondary level class, violating eligibility criterion 4.
- The abstract indicates the work is something other than a research paper.

At this stage of screening, only obviously ineligible papers were excluded. In ambiguous cases, the paper was retained for the next stage. 100 works were excluded for studying populations outside of post-secondary education (such as K-12 or graduate students), or for setting their study within a non-CS course.

While our initial search of the ACM Full Guide to Computing Literature had stipulated to only return works tagged as research articles, we discovered papers that did not match our own definition of those words. For our work, we wanted to focus on published research that sought to achieve some goal through collecting and analyzing data in a formalized way as established in Criterion 3. Seven additional works were excluded as they clearly did not fit into this definition. Thus, papers that were editorials, excerpts of blog posts, or summaries of working group meetings were excluded at this stage. In total, 107 works were removed at this phase, 436 papers remained.

Full Text Screening

Our first phase of full text screening focused on ensuring that all papers fulfil eligibility Criterion 6. We wanted to filter out papers that make only a passing reference to cognitive control. We also wanted to ensure all works in our corpus addressed the experiences of students learning CS at a post-secondary level. We first performed a vocabulary screening

on the full text of each paper, and then read through the methods section to learn about the study population.

To conduct the vocabulary search, we opened each article and searched for the relevant vocabulary to determine how frequently self regulation and metacognition were discussed in the text of the work. To do this, we searched for the roots of our wildcard strings "self-reg" (with and without a hyphen) and "metacog" in each paper. Only instances of this vocabulary in the text of the work were counted. Instances of these words in the works cited, tables, keywords, or abstract were not counted towards a paper's vocabulary total. If the terms appeared at least twice, the paper was included. A paper would also be included if it made exactly one reference to metacognition and one to self regulation. Any work where either metacognition or self regulation were only used once in the text of the study were discarded for violating Criterion 6. This led to 212 disqualifications.

One member of the research team read through each remaining paper to determine whether the methods met the qualifications laid out in Section 4.1.1. The population needed to consist fully of post-secondary students, or the presented data needed to be stratified such that the post-secondary students could be evaluated. Additionally, the course these students took needed to be a post-secondary CS course. Using this filtering process, 47 works were disqualified from our corpus. An additional 10 papers were removed when it became clear they were not research papers. These were things like editorials, blog posts, and magazine articles that were not caught as such during the Title and Abstract screening.

Seven works were categorized as 'unreadable', either because no full text version could be found or because no English translation could be found, violating eligibility Criterion 1. In total 276 works were disqualified at this stage, leaving 160 papers in our corpus.

Evaluating Depth

We then confirmed that each remaining paper had a sufficient focus on cognitive control. This was necessary in order to satisfy eligibility Criterion 6. To accomplish this, we read every remaining paper to determine if cognitive control was a focus of the work. Eligible papers either measured or observed some cognitive-control skill through the course of their study. This means a paper had to discuss cognitive control in the methods, results, or discussion sections. If a paper wrote extensively of a cognitive-control construct, but *only* did so as part of the background or introduction sections, it was removed from the corpus. This evaluation disqualified 45 papers from our corpus.

Following the work of Prather et al. [119], the research team chose to exclude papers that were less than two pages and papers that were works in progress as they could not examine cognitive control in sufficient depth. This disqualified four additional papers, leaving 121 in our corpus.

Critical Appraisal of Methods

The last step in creating our corpus was to conduct a critical appraisal of the methods. This was an integral step in the PRISMA review guidelines, to ensure each work in our corpus has a valid and trustworthy approach. In order to perform this assessment, we opted to use the Mixed Methods Appraisal Tool (MMAT) [116]. This tool was chosen as it offers support for methodologies that are common to education research like mixed method studies or quantitative studies on an un-randomized population. The 2018 version of the MMAT was used for this assessment.

This toolkit starts by evaluating all papers on two general standards. First, the MMAT asks: "Are there clear research questions?" Second, it asks: "Do the collected data allow to address

the research questions?". Papers that qualified were then evaluated on five methodology-specific standards. These centered on ensuring that data were collected, reported, and synthesized according to the standards of the methods in question. Studies that were mixed methods in nature were evaluated using the standards for all relevant methodologies, and on a separate mixed-method specific criteria list. 57 works were removed for not fulfilling the MMAT toolkit standards.

38 of those 57 (66.66%) were removed for not having a research question or other stated goal for the work. This does mean about half of the remaining pool of papers was removed due to the MMAT. However, as mentioned previously, several studies were removed during the title and abstract screening and at the full-text screening phases for not fitting within our definition of research. It is possible the ACM-DL's labeling of what is and is not a research paper is inaccurate (or just wider than we expected). Many of these papers were not necessarily weak, but were experience reports or presentations of software rather than research attempting to answer a question.

16 literature reviews were also disqualified as they violate eligibility Criterion 6. The most relevant of these have been highlighted in Section 2.

After completing these steps, 73 papers were removed and we were left with 38 research studies. The full list of included works can be seen in Appendix B.

4.1.4 Data Extraction

Once the final corpus of works was created, the next step was to extract the relevant data. One trained member of our research team read over every study and made note of the relevant passages. We used the following criteria to extract data that was relevant to answering our research questions:

To be relevant to RQ1, a passage needed to make it clear what they considered to be a cognitive-control skill, or be related to an already clearly defined skill. For example, "Learners may self-regulate by being aware of limitations in their knowledge of related problems" is a passage from Loksa and Ko's study [81]. Here, awareness of limitations in knowledge is explicitly mentioned as an aspect of self regulation. After this key statement, any sentence going into further detail describing this form of awareness was also extracted. Similarly, authors often created tables listing a number of self-regulation skills, like those seen in Lyon et al. [85], Prather et al. [121], or Falkner et al. [39]. All skills from such a list were extracted, as were any sentences from the text of the paper that further described the listed skills.

Our focus is on cognitive-control skills studied in CS education research. We only extracted passages when authors clearly were drawing a connection between cognitive control and a skill. This criterion was chosen because it helped reduce subjective judgments when extracting passages. Inferring whether a vague passage was indeed talking about metacognition or self regulation had the potential to bias this set of passages to the author's notions of what did or did not count. Thus, if a paper never drew a clear connection between a skill and cognitive control, no statements were extracted about that skill.

A passage could be of any length, though ultimately none were longer than a paragraph. Four authors were involved in qualitatively coding these passages and creating the list of skills and indicators discussed in Section 4.2. The process of qualitative coding is described in more detail in Section 4.1.5.

In total, 188 passages were extracted for qualitative coding to answer RQ1.

To answer RQ2, one member of the research team read over each paper again and made a determination if the **paper's outcomes claimed to connect some aspect of cognitive**

control to academic success. In order to reduce bias, this was done by noting at least three passages from each paper. First, at least one passage from the original collection of 188 passages had to note that a particular concept was an aspect of cognitive control. Second, the paper must contain at least one passage (not necessarily in our collection of 188 passages) demonstrating that the authors were connecting cognitive-control constructs to success. Third, the paper must contain at least one passage (not necessarily in our collection of 188 passages) describing the relationship observed. This allowed us to note when the skills and indicators from RQ1 were connected with success, and gauge the significance of that connection.

Importantly, no study was discounted if the examined skill did not have a measurable effect on success. The only requirement was that a work clearly connected at least one cognitive-control skill to academic success, not that the data supported that connection.

Qualitative coding was not performed for passages extracted to answer RQ2. Rather, these passages were used to write a short summary of each finding reported by a paper. This helped the research team better understand how a skill related to academic success in each paper so overall trends could be presented here. A spreadsheet containing all of this information was created and is discussed in more depth in Section 4.1.6.

Extraction From MSLQ

As mentioned in Section 2.1.2, eight papers (21%) of the final corpus sought to measure cognitive control through Pintrich's widely used MSLQ inventory [14, 15, 32, 75, 76, 79, 121, 139].

In this inventory, students rate how true 81 statements are to their lives and learning habits. Not all of these questions were extracted and used for answering RQ1, however, as several

subscales focus on measuring related constructs like motivation or critical thinking, rather than self regulation or metacognition. Following our general policy for extracting data for RQ1, we wanted to capture what the authors of each study thought were cognitive control skills. In some cases, the authors were clear about what MSLQ subscales they used to study self regulation. Duvall et al.noted that they used both the Metacognitive Self-Regulation (MSR) subscale and the Effort Regulation subscale to measure self-regulation skill [32]. Leppänen et al.noted they used three subscales (MSR; effort regulation; and time and study environment management) to measure self-regulation skills [75]. When studies made no mention of which subscales they used to measure self regulation, it was assumed the MSR subscale was used as it most closely matched our working definition.

Multiple versions of the MSLQ were used by papers in the corpus. Most used the MSLQ from 1991 [114], which we will refer to as MSLQ-1991. This makes sense as it is the most well validated and most commonly used outside of CS education work as well [112, 126].

Castellanos et al.used MSLQ-Columbia [124], which is a Spanish translation of MSLQ-1991. It also reorganizes the MSR subscale into three smaller metacognition-focused subscales, using 9 out of 12 questions from the MSLQ-1991 MSR subscale. These smaller subscales are: Planning, Monitoring, and Study Method Regulation [124]. Due to resource limitations, the original Spanish text could not be coded for this work. Instead, Ramírez-Echeverry et al.'s work [124] discussing the translation process was referenced. No lexical changes between the MSLQ questions and MSLQ-Columbia questions were reported by Ramírez-Echeverry et al. Therefore, the coding team for this review coded the MSLQ-1991 in English and noted where a passage was translated for use in the MSLQ-Columbia.

The slightly older MSLQ-1990 was used in two studies. This is similar to MSLQ-1991, but all questions are slightly rephrased. Thus, all questions from the subscales detailed in Table 4.1 were extracted as unique passages and coded independently to help answer RQ1.

Citation	MSLQ Version	Subscales Extracted
[32]	MSLQ-1991	Metacognitive Self-Regulation Effort Regulation
[76]	MSLQ-1991	Metacognitive Self-Regulation
[15]	MSLQ -Columbia	Planning Monitoring Study Method Regulation
[121]	MSLQ-1991	Metacognitive Self-Regulation (3 questions)
[75]	MSLQ-1991	Metacognitive Self-Regulation Effort Regulation Time and Environment Management
[14]	MSLQ-1990	Metacognitive Self-Regulation
[79]	MSLQ-1991	Metacognitive Self-Regulation
[139]	MSLQ-1990	Metacognitive Self-Regulation

Table 4.1: Version and subscales extracted from each paper that used the MSLQ

4.1.5 Data Synthesis

After the data extraction phase was complete, the research team performed qualitative coding on all 188 passages that identified a cognitive-control skill or behavior. Each passage was given one of 33 codes describing cognitive-control skills, behavioral indicators of those skills, or situations where students had difficulty with using those skills.

General Process: Passages were assigned to different members of the coding team such that at least two coders reviewed each passage. The process of assigning passages will be

described after going over the process the team used to assign codes.

To start the process, a team member would review the collection of passages assigned to them, each within the context of its originating paper. The team member would then write short summaries for each passage. In qualitative analysis, this process is called 'memoization', and it helps a coder gain familiarity and identify trends with all passages before starting to assign codes.

Next, each team member would independently assign a code to each passage. The team would then convene to discuss how each team member had coded each passage. Team members had the opportunity to change what code they had assigned if they felt another team member's code was a better fit. If the two coders could not reach an agreement, a third member was asked to review the code in context and assign their own code. In most cases, the tie-breaker reviewer would agree with one of the codes that the two original reviewers had suggested. In rare cases, the tie-breaker reviewer would chose a third code that they felt better fit the passage. The three coders would then meet, return to the paper, and reread the passage in context together. They would then come to a determination that all three could agree on.

Work Assignment: There were four members of the coding team overall, but they never worked together all at once.

Three reviewers coded the first 99 passages. The team of three was initially assigned 20 passages and followed the process described above as a group in a calibration phase. This exercise helped all team members gain a sense for the coding process, calibrate understanding of what the preliminary codes meant, and identify where new codes might be needed.

Following the calibration phase, the next 79 passages were memoized, coded, and discussed.

A code was assigned to a passage if at least two out of the three members of the coding team assigned the same code to it.

After all codes had been assigned to these 99 passages, the team structure shifted. One member of the original coding team was unable to continue working, while another had reduced availability. To address these constraints, a fourth team member was added and assignments for the remaining 89 passages were adjusted. All passages were assigned such that they received an initial code from at least two team members. As with the first round of coding, the three team members initially worked together in a calibration phase to assign codes to 24 passages to calibrate their understanding. Following that, the same process of memoization, coding, and discussion was used to complete the process.

Agreement: Table 4.2 describes the percent agreement between all pairs of coders that worked together. The team member labelled 'R3' is the one who could not participate in coding the second half of the passages. 'R4' represents the new team member who took their place. Since R3 and R4 never coded a passage together, they have no agreement to report.

Percent Agreement Before Discussion

	R1	R2
R1		
R2	62.33%	
R3	52.17%	30.43%
R4	81.82%	83.33%

Percent Agreement After Discussion

	R1	R2
R1		
R2	72.08%	
R3	66.30%	41.30%
R4	80.56%	76.67%

Table 4.2: Percent agreement between all pairs of coders

Final Review: After each passage had been assigned a code, the coding team met once more to review and discuss if all passages had been assigned the best possible code. The list of codes and corresponding assigned passages went through considerable refactoring at this stage as trends in skills and behavioral indicators emerged.

Passages were then reassigned to these refactored codes based on where they fit best. For example, "Forming the wrong conceptual model about the right problem" was a passage from Prather et al. [117]. Initially this was coded as *Task Analysis*. However, in this refactoring phase, it became clear there was a trend of passages more specifically discussing situations where students struggled to effectively analyze a task. Therefore, the coding team created the sub-category: '*Task Analysis* - difficulties' and reassigned some passages to this new sub-category.

Skills were also grouped together into overall skillsets that loosely related to when in the work process that students might use them. Section 4.2 describes the identified skillsets in more detail.

After this refactoring, the remaining three members of the coding team reviewed all passages, discussed the new assignments, and reached 100% agreement on the assignment of codes to all passages.

Use of Theory-Generated Codes: Before starting to code, all members of the coding team were given an overview on the basics of cognitive control by reading through three works to get an overview of the important ideas. To do this, all team members read over Loksa and Ko [81], Prather et al. [119], and Garcia et al. [45].

Initially, the codes for these passages were created inductively, meaning the team read through passages and created their own codes and definitions to suit the text. However, it became clear even after training that many of the codes created at this stage were ill-defined and did not have a good mapping to existing cognitive control theory.

In order to effectively answer RQ1, we wanted to ensure that the skills we identified had a solid grounding in existing theory. We therefore introduced the list of skills from our

Delphi Process [30] as a list of theory-generated codes to get started. In this context, a *theory-generated code* is one derived from existing literature, and is distinct from an *in-vivo* code, which is created by reviewers while coding a text [27, 89]. The skills from our previous Delphi Process study were selected as the set of terms to start with as they define a set of cognitive control skills that each have a firm grounding in both educational theory and existing study within CS education research.

Theory-generated codes act as a way to provide a consistent and reasonable starting point for qualitative coding, but are never meant to take the place of codes derived from the text. Members of the coding team were strongly encouraged to see these initial codes as optional. They were directed to create new codes in situations where none of the theory-generated codes seemed appropriate. They also had the option to mark any passage as "unsure" if they wanted additional information about existing theory before assigning a code.

The coding team took these directions to heart and did not simply use the list of skills from the Delphi process stidy [30] to deductively code data. They added many new codes and removed unhelpful theory-generated codes. In the final review phase discussed in Section 4.1.5, the coding team also grouped these skills differently from the Delphi Process. While we believe the use of these theory-generated codes as a starting point was an appropriate choice for answering RQ1, this approach had an undeniable impact on our answer to this question. A more in-depth discussion on the biasing effects of this decision and the ways we have further sought to address bias is addressed next.

4.1.6 Areas of Potential Bias and Artifacts of this Work

We seek to be as transparent as possible about areas of potential bias within this work. To that end, this section describes places where bias was potentially introduced into our work.

We also provide publicly available artifacts that document our process. In doing this, we hope that future researchers will be able to build upon our work.

All of the additional artifacts described in this section can be found in in supplementary documents to this dissertation.

Study Selection Bias: Bias was likely introduced in searching and selecting studies for this corpus.

While the ACM Full Guide to Computing Literature online library is extensive in breadth, it does focus on *computing* literature. Thus, our resulting corpus focuses on only the works published at some computing-focused venue. This was preferable as it helped us more effectively identify research that studied the experience of post-secondary CS students. However, it is possible relevant papers were missed in our initial search because they were published in related domains outside of computing, like education research.

Throughout the process of study selection, we emphasized the use of certain vocabulary words. Our search string was formed to look for variants of 'self regulation' and 'metacognition', and our full text screening disqualified papers that did not use these terms enough. Hypothetically, a highly valuable study on help-seeking that made no mention of cognitive control, would not be returned in our initial search. Thus, the results presented here reflect the state of cognitive-control research, not the state of the art on any individual skill.

Subjectivity was likely introduced when conducting the critical appraisal of bias. The first question within the MMAT toolkit is: *Are there clear research questions?*. This may seem straightforward, but there were cases where only examining a paper with research questions would have been overly limiting. For example, Herman et al. [57] state: "The goal of this paper is to understand the ways in which students fail to translate accurately from English

to Boolean expressions". Even though it is not phrased as a question, the research team read this as a clear statement of research aims and the approach taken supports this goal. Such determinations inherently involve some level of subjective judgement.

Other questions from the MMAT like "Do the collected data allow to address the research questions?" or "Are findings adequately derived from the data?" introduce subjectivity as well. A random sampling of these decisions was validated by a second researcher (with 100% agreement), but it is possible that the paper selection process introduced some level of bias within our work. With this in mind, we have created a document detailing every paper in our corpus (after the de-duplication stage) as well as details about when and why it was excluded.

Overall, we feel we have constructed the best possible corpus given our constraints. Our results demonstrate that a diverse set of skills are being examined in cognitive control research. If a study was indeed incorrectly included (or excluded), it might impact our observations about which skills are seeing a lot of focused study, but it would not have much effect on which skills were observed.

Data Extraction Bias: When extracting passages for RQ1, we attempted to reduce subjective decision-making while extracting data by only identifying skills clearly called as self regulation or metacognition. While we chose to do this to reduce subjective decision-making, there was a limit to how effectively we can eliminate judgment calls from this process. As described in Section 4.1.4, we sought to identify passages where authors explicitly noted a skill was an aspect of metacognition or self regulation. Still, cases where it was unclear what skills a study was referring to did come up.

For example, Peteranetz et al. [109] outline four aspects of "student self-regulation and strategic engagement" in their work. Those aspects are: "general metacognitive self-regulation",

"Knowledge Building", "dysfunctional self-regulatory strategies", and "behavioral engagement with the class" [109]. While the term self regulation is directly in two of these elements, it is unclear if the authors see *Knowledge Building* or behavioral engagement with the class as a component of self regulation, a component of strategic engagement, or a component of both ideas. Ultimately, it seemed safest to cast the widest possible net and extract passages relating to all four concepts. While we feel that this was most appropriate for answering our research questions, it is possible Peteranetz et al.defined self regulation more narrowly. While such judgements were not common, they were a component of this review. Generally speaking, we did our best to read closely and extract everything that seemed like it could be related. However, it is possible we have misunderstood the author's understanding of cognitive control and included constructs that are related but external to their definitions of self regulation or metacognition.

To be as transparent as possible about this inevitable form of bias, a spreadsheet detailing every extracted passage to answer either research question is available as supplementary documents to this dissertation.

Bias within Data Synthesis: Another potential area of bias was the decision to provide the coding team with the list of skills from the Delphi Process Study [30] as a set of theory-based codes. Had the coding team worked solely from the extracted passages, it is likely the resulting list of skills would be different. However, we believe it is also likely such a list would carry a much weaker basis in existing theories of cognitive control, making it a much weaker list overall. Thus, while our methods are sufficient to answer our research questions, it did mean our coding team was likely biased in creating this particular list of skills.

To be as transparent about how our coding team utilized the list of skills from the Delphi Process Study, we have created a document that serves as a change log. This document

details how our theory-based codes developed into our final set of codes. The diagram of the final set of codes presented in this research can be seen in Figure 4.3.

Within the publication version of this work, we included figures that offered a side-by-side comparison of the skill hierarchy from the Delphi Process study and those created from this review of the literature. Please refer to Figure 3.1 for hierarchy created from the Delphi Process Study and to Figure 4.3 for a comparison.

Language Bias As the author screening papers and extracting data only speaks English, only papers originally written in English or with a translation to English could be included in this work. Ultimately, only a handful of works were excluded for this reason, but a survey of a more diverse set of studies in a wider range of languages could be a valuable area for future work.

4.2 Identified Skills

This section provides a discussion of the 11 cognitive-control skills and 10 behavioral indicators of a skill in action that were identified from the study corpus. Specifically, these skills and behaviors where identified from analysis of the 188 passages extracted from our corpus and coded as described in Section 4.1.5. Recall that each was given one of 33 codes describing cognitive-control skills, behavioral indicators of those skills, or situations where students had difficulty with using those skills.

As described in Figure 4.3, the identified skills are mostly grouped into four overall skillsets. To keep things clear, skillsets will be in bold while skills will be italicized.

Table 4.3 presents our working definition for the skills along with the behaviors used to

Code	Definition
Planning (skillset)	General discussion of 'strategizing', typically as a first step to completing a task
Task Analysis (skill)	Forming an accurate conceptual model of the task at hand
Task Analysis - difficulties	Cases of students who formed an inaccurate conceptual model and their subsequent difficulties completing tasks
Task Analysis - Reinterpreting Materials (indicator)	Students understanding a problem by transforming or annotating assignment text
Scheduling (skill)	Students intentionally allotting blocks of time in the future to complete the task
Scheduling - difficulties	Cases of students not setting aside enough time to complete work before a deadline and their subsequent difficulties completing tasks
Scheduling - Starting Early (indicator)	Students scheduling by allocating time blocks to work that are well before an assignment is due
Scheduling - Spacing (indicator)	Students scheduling by allocating many smaller time blocks throughout the course of an assignment (rather than allocating one large block for all work)
Decomposing (skill)	Students taking an abstract task and breaking it into smaller more concrete sub-tasks as a way to construct an overall strategy or algorithm
Decomposing - difficulties	Cases of students who did not break a task into smaller components and their subsequent difficulties completing tasks
Decomposing - Outlining (indicator)	Students decomposing a problem by creating method skeletons before implementing
Decomposing - Enumerating (indicator)	Students decomposing a problem by creating a list of sub-tasks needed to complete the larger goal
Monitoring (skillset)	General discussion of 'maintaining an awareness of self'
Monitoring Correctness (skill)	Students maintaining an awareness that the work they are doing is correct
Monitoring Correctness - difficulties	Cases of students who were unaware that the work they were doing was incorrect and their subsequent difficulties completing tasks
Monitoring Correctness - Predicting Confidence (indicator)	Observing a student's awareness of their correctness by having them self-report their confidence in their answers
Monitoring Correctness - Validating (indicator)	Students monitoring their correctness by taking steps to prove to themselves that the answer they came up with is correct
Monitoring Progress (skill)	Student maintaining an awareness of how close they are to completing their task
Monitoring Progress - difficulties	Cases of students who were unaware of how close they were to finishing and their subsequent difficulties completing tasks
Emotional Regulation (skill)	Students maintaining an awareness of their current emotional state and the potential impact those emotions could have on their work
Emotional Regulation - difficulties	Cases where students were not mindful of their emotions and their subsequent difficulties completing tasks
Adaptation (skillset)	General discussion of students making a strategic change to their work
Help-Seeking (skill)	Students seeking outside assistance after recognizing a need for that help.Involves assessing what questions to ask, what resources to seek help from, and when to reach out for help
Reducing Distractions (skill)	Students making an effort to adjust their study environment so they do not become distracted
Reflection (skillset)	General discussion of students considering their approach to completed tasks (or subtasks) as it relates to work they will do in the future
Reflection - Reflection Using Internal Standards (skill)	Students assessing their work in the context of their expectations of themselves
Reflection - Reflection Using External Standards (skill)	Students assessing their work in the context of course expectations
Knowledge Building (skill)	Students engaging in in ways that are not for credit within a course
Knowledge Building - Practicing (indicator)	Students building knowledge by working on ungraded practice exercises
Knowledge Building - Investigating (indicator)	Students building knowledge by seeking out answers to questions not directly related to the course
Procrastination (general indicator)	Observations of students who appear to intentionally delay working in spite of the negative consequences of that delay.

Table 4.3: All Codes and Definitions

indicate a given skill. Table 4.4 shows references to all papers that discuss a given skill, behavior, or difficulty.

For ease of presentation, our skills and indicators are organized into a hierarchy, with related skills collected into skillsets. Our skillsets match the three common phases of self-regulation models identified by Puustinen et al. [123]. These phases are the preparatory phase, the performance phase, and the self-control phase. All three major theories of self regulation discussed in Section 2.1.2 have analogues Puustinen et al.'s phases.

In the preparatory phase, students set intentions and strategize. We represent this with our **Planning** skillset where *Task Analysis*, *Scheduling*, and *Decomposing* are discussed. In the

Code	Code Count	Referenced In
Planning (skillset)	4	[15, 39, 109, 136]
Task Analysis (skill)	2	[39, 75]
Task Analysis - difficulties	2	[39, 75]
Task Analysis - Reinterpreting Materials (indicator)	6	[3, 25, 39, 117]
Scheduling (skill)	10	[15, 39, 85, 103, 107, 121]
Scheduling - difficulties	2	[3, 39, 75]
Scheduling - Starting Early (indicator)	6	[3, 26, 62, 74, 162]
Scheduling - Spacing (indicator)	4	[3, 18, 62, 74]
Decomposing (skill)	15	[14, 15, 32, 39, 75, 76, 79, 81, 85, 109, 121, 130, 139]
Decomposing - difficulties	1	[3]
Decomposing - Outlining (indicator)	7	[39, 125, 145]
Decomposing - Enumerating (indicator)	3	[15, 32, 39, 75, 76, 79, 103]
Monitoring (skillset)	3	[57, 109, 143]
Monitoring Correctness (skill)	21	[15, 24, 32, 39, 57, 73, 76, 81, 103, 118, 132, 151] [14, 75, 79, 139]
Monitoring Correctness - difficulties	6	[3, 57, 117]
Monitoring Correctness - Predicting Confidence (indicator)	5	[24, 73, 132, 143, 151]
Monitoring Correctness - Validating (indicator)	14	[15, 32, 39, 75, 76, 81, 85, 103, 125, 145] [14, 79, 139]
Monitoring Progress (skill)	3	[75, 81, 107, 145]
Monitoring Progress - difficulties	2	[117]
Emotional Regulation (skill)	5	[32, 39, 75]
Emotional Regulation - difficulties	5	[14, 15, 121, 139]
Adaptation (skillset)	4	[15, 32, 39, 75, 75, 76, 79, 143]
Help-Seeking (skill)	11	[3, 28, 39, 85, 109, 125]
Reducing Distractions (skill)	4	[15, 39, 75, 121]
Reflection (skillset)	3	[13, 58, 85]
Reflection - Reflection Using Internal Standards (skill)	2	[3, 15, 32, 75, 76, 79]
Reflection - Reflection Using External Standards (skill)	3	[32, 75, 76, 79, 144]
Knowledge Building (skill)	1	[109]
Knowledge Building - Practicing (indicator)	4	[14, 39, 85, 121, 139]
Knowledge Building - Investigating (indicator)	3	[39]
Procrastination (general indicator)	11	[3, 35, 39, 107, 130, 162]

Table 4.4: Code Counts and References

performance phase, students self-observe and control their behavior, which we divided into two distinct skillsets. Self-observation is reflected in our **Monitoring** skillset where students maintain an awareness about their correctness, completeness, and emotional state. Self-control is reflected in our **Adaptation** skillset where students adjust their approach based on their observations by seeking help or regulating their environment to reduce distractions. In the appraisal phase, students react and reflect. This is encompassed in our **Reflection** skillset where students think back on the work they have done and make larger-scale plans for the future.

One skill and one indicator did not fit into any of these four skillsets or three phases, as they could occur at a number of different points in the process of preparation, performance, and appraisal. *Knowledge Building* could occur while a student is preparing to work, while working, or after finishing. Similarly, procrastination is marked as a 'generalized indicator' because it is a behavior that might result from difficulties associated with many skills. These two cases are discussed separately from the other skills and indicators on this list.

In Table 4.4, references never represent a super set of other codes. For example, the two papers listed in the second row, *Task Analysis*, discussed this skill. It so happens that both of those papers also discuss difficulties with *Task Analysis*, so they are also listed on the third row. Of the two, only one paper, Falkner et al. [39], also discusses reinterpreting materials as a part of *Task Analysis*. Some papers contain passages that were coded as a skillset, rather than any specific skill or indicator. These represent situations where, even within the context of the paper, a skill from our list was never clearly defined. For example, Peteranetz et al.list out their relevant components of self-regulated metacognitive strategy, but do not elaborate on what they mean by 'planning' [109]. It could refer to a student determining *how* they intend to approach a problem, but it could also refer to *when* a student intends to work. All we can conclude is that the authors of this work identified some planning processes as an important component of cognitive control. The best fit for this passage was, then, the planning skillset rather than a specific skill.

Figure 4.3 shows a diagram of all identified skills, indicators, and difficulties grouped by skillset.

MSLQ The Motivated Strategies for Learning Questionnaire is an 81-question inventory divided among 15 subscales. Since only some of these subscales were used to measure cognitive control, only the questions from relevant subscales were extracted for analysis in this

study. A single extracted subscale from the MSLQ could cover a wide variety of the skills listed in Table 4.3. This makes sense as the MSLQ was designed to capture a holistic view of a student's ability to self regulate. However, it does make reporting the skills that the MSLQ captures somewhat difficult as many papers use different versions of the inventory, different subscales, or even subsets of questions within a subscale. Table 4.5 describes the skills that are captured in various forms of the MSLQ and references the works within this corpus that use this inventory directly.

We next discuss the various skills in the order that they are presented in Table 4.4.

Skill	Metacognitive Self-Regulation (MSLQ-1990)	Metacognitive Self-Regulation (MSLQ-1991)	Effort Regulation (MSLQ-1991)	Time and Environment Management (MSLQ-1991)	Metacognitive Planning (MSLQ-Columbia)	Metacognitive Monitoring (MSLQ-Columbia)	Metacognitive Study Method (MSLQ-Columbia)
Task Analysis							
Scheduling				[75]			
Decomposing	[14, 139]	[32, 75, 76, 79]			[15]		
Monitoring Correctness	[14, 139]	[32, 75, 76, 79, 121]				[15]	
Monitoring Progress				[75]			
Emotional Regulation	[14, 139]	[32, 75, 76, 79, 121]	[32, 75]			[15]	
Help-Seeking							
Reducing Distractions				[75]			
Reflection		[32, 75, 76, 79, 121]		[75]			[15]
Knowledge Building	[14, 139]						

Table 4.5: Skills Captured Within Different Versions of the MSLQ

4.2.1 Planning Skills

The corpus contains many skills that fall under the umbrella of **Planning**. The term was used to describe how a student understands a problem, strategizes their approach, and allocates time. The skills in Table 4.4 associated with this skillset reflect those three components. These skills all loosely fit within the *preparatory* phase of Puustinen et al.'s taxonomy [123]. Though a student may call upon them while working, they are most likely used before beginning to work in order to set intentions and strategize.

Task Analysis

Task Analysis is often discussed within this corpus through the lens of starting too quickly, though the skill does extend beyond this one behavior pattern. Novice programmers commonly demonstrate an eagerness to start writing code before they've fully thought things through. Denny et al.note: "...students who misinterpret a problem statement and don't have the metacognitive awareness to identify when their mental model deviates from that statement may struggle to make progress towards a working solution" [25]. Ultimately, *Task Analysis* concerns the critical practice of taking time to form a proper mental model of the task. This prevents misunderstanding requirements, which can ultimately cause students to get stuck.

While a correct conceptual model is the ultimate goal of *Task Analysis*, a student cannot actually *know* if their model is indeed correct. Therefore, correct knowledge of requirements is not an aspect of cognitive control by itself. What is critical is that a student thoughtfully and intentionally reads and processes instructions. If they do this and do not ultimately come up with a perfect understanding of their task, they have still engaged in *Task Analysis*. For example, if a student thoughtfully and methodically reads over instructions but misunderstands how a feature of their software should work, they still applied their skill at *Task Analysis* to the problem.

Within general education research, *Task Definition* is a component of the Winne-Hadwin model [157] and Zimmerman's Cyclical Phase Model [165]. Greene et al.highlight a trend of implicit assumption among instructors that problems have been written in a way that students will understand [49]. Malmberg et al.noted "empirical research has shown that learners do not always completely understand what the teacher or instructor expects for task accomplishment." [88]. Both authors highlight how, in order to succeed, students need to

be able to identify metacognitive cues from a problem statement that help them understand what they are being asked to do. They also need to be able to synthesize such cues with their own perception of the task.

Seeing a student reinterpret materials is one indicator that has been used to identify CS students taking time to form that good conceptual model. For example, seeing a student review [85, 103] or annotate [121] a problem statement would be indicative of *Task Analysis* in action.

Researchers have observed during Think-Aloud studies that CS students struggle with *Task Analysis*. The authors of these works will frequently comment on times when students verbalized incorrect conceptual models [3, 117, 118, 121]. Prather et al.found formation of a correct conceptual model was the "most glaring inconsistency" between students who successfully completed their Think-Aloud assignment and those who did not [117]. In their code book for their Think-Aloud experiment, Loksa and Ko identified two types of "Comprehension Monitoring". One of these types focused on "participants absorbing information, often from examples or when attempting to understand a problem" [81]. This was aggregated with another form of Comprehension Monitoring that focused more on participants verbalizing that they did not understand their work, or explaining an idea to themselves. This second form was coded as *Monitoring Correctness* and will be discussed more in Section 4.2.2.

Scheduling

With 22 extracted passages from 12 different papers, the intentional optimization of time used to work was a frequently discussed skill in this corpus. In the context of the **Planning** skillset, *Scheduling* focuses on how students allocate time to work. It requires a student to assess the work they have to do and apply metacognitive knowledge to estimate how long

they will need to complete it. They also must make judgements about the length and number of work sessions they will need to complete a task successfully.

Within this corpus, allocating time was often highlighted as important [3, 15, 26, 39, 85, 103, 107, 121], but few studies sought to define or explore the nuances of *Scheduling* outside of examining its indicators. For example, Arakawa et al.used indicators of *Scheduling* to identify students struggling to self-regulate, but did not expand into greater detail about *Scheduling* as a skill [3]. This skill is frequently highlighted as an important aspect of cognitive control. This corpus identified starting early and spacing as two potential indicators of *Scheduling*.

Starting early is summarized well in Zhang et al.'s 'Days Started Before Due' metric [162]. This indicator describes the difference between when a student starts an assignment and when it is due, and is a commonly used indicator of *Scheduling* [26, 62, 74, 162]. The metric 'days between starting and submitting the final product' is most frequently employed for longer-term assignments, typically with at least a week between assignment date and due date.

'Spacing' is used to describe a student distributing work over a period of time. Leppänen et al.summarize the educational psychology behind this behavior: "students who study the same set of material for the same overall time tend to perform better in tests if the studying is done in multiple spaced chunks instead of in a single session" [75]. As indicators, starting early and spacing are grounded in a significant body of general education research. Macan et al.noted in 1990 that "allocating time properly or last-minute cramming for exams, have been frequently discussed as a source of stress and poor academic performance" [87].

Decomposing

Decomposition helps students determine an overall approach and facilitates better prioritization and allocation of time. The need to break down a task into smaller tasks or set intermediate goals for one's self was a frequently discussed skill. Indeed, there were 26 extracted passages from 17 different papers, making this skill second only to *Monitoring Correctness* (Section 4.2.2 in its frequency. As described previously, the term 'planning' meant different things in different studies. However, this idea of working to create sub-goals was the most common meaning of the term 'planning'.

There is some evidence in education research to indicate that novices Decompose in a qualitatively different way from experts. Ho[59] noted that experts tended to be more explicit and deliberate in how they decomposed a problem, while novices were not so structured in their behavior. This is not to say novices did not decompose. Instead, Ho noted novices still appear to have performed *some* problem decomposition based on analysis of the work produced [59], but appeared to do so internally. He divides these two styles of decomposition into 'explicit' forms, more frequently used by experts, and 'implicit' forms more typically employed by novices. Thus, creating observable artifacts may serve as both measurement and intervention.

Since this internal form of *Decomposition* has no behavioral indicators, it is perhaps unsurprising that the two common indicators identified in this corpus involve asking students to perform more explicit decomposition. This was commonly done by looking for some document created as an artifact of a student's decomposition process. Many studies observe that while programming, students might outline code by making method skeletons or other design artifacts before starting to program [39, 125, 145]. The other common indicator was seeing students enumerate a list of "concepts, stages, or components'' [15]. This could be

done by writing out a to-do list or verbalizing a set of goals [32, 39, 75, 76, 79, 103].

4.2.2 Monitoring Skills

Monitoring skills focus on different variables that students must maintain an awareness of. Within this corpus, we identified that many works commented on a student's need to monitor and validate the correctness of their work as well as their progress, given the allotted time. Another emergent theme we saw was that students should stay aware of their emotional state, or at least manage frustration and boredom as necessary to succeed.

Within Puustinen et al.'s framework, these skills would all be classified as forms of 'self-observation', which is part of the *performance* phase. The ways students act on that awareness were classified as forms of **Adaptation** and will be discussed in Section 4.2.3.

Monitoring Correctness

With 46 passages from 23 different papers, *Monitoring Correctness* is the most frequently discussed of our skills. Many studies call this idea "metacognitive accuracy'' and define it as "the relationship between confidence and correctness'' [132]. Both definitions capture the same idea: students need to regularly check in with themselves to make sure they understand what they are doing and are aware that their answers are correct.

In Think-Aloud studies, validation could be as simple as a student explaining to themselves why a choice is right [81, 103, 121, 125, 145]. Statements like "this was the right loop condition because it halts at the end of the list'' [125] indicate a student is validating their choice to themselves. In a Think-Aloud study on Boolean logic, Herman et al.noted that many students did not notice mistakes in their work and only a handful validated their answers by translating the logic back to English [57]. Loksa and Ko looked for such behaviors

when they observed students "...evaluating how well their implementation solves the problem, usually by testing and debugging'' [81].

Outside of settings where a researcher could observe validation directly, studies have mentioned looking for evidence of testing code [15, 39, 81, 85, 145], though few studies do more than note its importance. Automated Assessment Tools (AATs) offer support when looking for testing behavior. Arakawa et al.used "fail[ing] to pass the same test case in three sequential commits & pushes to GitHub'' [3] as one of three indicators that a student might be struggling to self-regulate.

Another common way of observing this skill is to ask students to *Predict Confidence* immediately after finishing a test. While currently limited to just exams, these do a good job of measuring a student's *perception* of correctness and can be easily compared to the actual correctness of the work done. As these confidence predictions often take the form of a straightforward Likert-scale inventory, this form of measurement is easy to implement and has seen a great deal of study [24, 25, 73, 132, 143, 151]. Such predictions have a well-established basis in Psychology research. Hart reported in 1965 using "feeling-of-knowing" as a measure for what is and is not in memory [56], so it is unsurprising that this indicator shows consistent positive correlations with success across nearly every study that employs them.

Beyond 'feeling of knowing', *Monitoring Correctness* as a whole has been a well-known aspect of cognitive control for decades [106]. Baker and Brown described "monitoring the effectiveness of any attempted action" as an important metacognitive strategy in 1984 [6], and that is echoed in more recent research as well by Greene et al. [50]. However, Winne et al.have demonstrated that novices, overwhelmed with information, have difficulty with monitoring correctness effectively [159].

Monitoring Progress

A student's sense of where they are in solving a problem was another aspect of Monitoring discussed in this corpus. Loksa and Ko highlight 'Progress Monitoring' in their framework of self regulation saying: "Programmers who explicitly monitor their progress toward solving a problem are more successful...The more learners monitor when a task is complete, the more successful they should be" [81]. Students need to have a metacognitive standard about how much time they need to devote to a piece of work and stay aware of their current pace. Students must then make adjustments to their schedule and behavior using that knowledge and awareness.

Prather et al.identified two key ways students misunderstood their progress through an assignment. Students associated seeing AAT feedback with being mostly finished with an assignment, even when there was still much work to be done. This assumption led students to have a "poor conception of location in the problem-solving process" [117]. Besides simply being unaware, Prather et al.also observed this hindered student's proficiency with questioning if they were on the right track as they had an "unwillingness to abandon a wrong solution due to a false sense of being nearly done" [117]. Thus, *Monitoring Progress* appropriately is important not only because it facilitates better allocation of future time, but also because it helps maintain other forms of awareness more effectively.

Ilves et al.scaffolded this form of awareness by showing students metrics about how they used their time compared to the class averages [62]. While this helped students practice staying aware of time, they did not use any behavioral indicators to judge how students improved over time. Indeed, outside of observing students directly in a Think-Aloud study setting, no works within this corpus identified behavioral indicators of *Monitoring Progress*.

Emotional Regulation

This skill is characterized by staying aware of one's emotional state and finding ways to succeed while taking one's emotional state into account. The passages that were assigned this code most frequently originated in the MSLQ. Many statements in this inventory focus on students continuing to work through disinterest or boredom. This could be seen through statements like: "even when study materials are dull and uninteresting, I keep working until I finish'' [110]. Other questions more directly attribute disinterest to affect-related reasons like: "I often feel so lazy or bored when I study for this class that I quit before I finish what I planned to do'' [114]. In either case, the statements focused on maintaining energy to work when experiencing a demotivating emotion.

There were also some questions which focused on more than just disinterest or boredom. These extended into resilience towards challenge and failure. The 1991 Effort Regulation subscale includes the statement "When course work is difficult, I give up or only study the easy parts'' [114]. Here, it is unclear if a student is experiencing anxiety, frustration, laziness, or some other emotion entirely. However, it does appear that some emotion is influencing their behavior.

Falkner et al.'s 2014 case study was the only work not using the MSLQ to discuss regulating emotions. In their work, they note "avoiding sources of anxiety'' was a form of personal management that students recognized was important for learning CS. As this phrase had no additional context, it is unclear if this was brought up as a form of effective *Emotional Regulation* or ineffective.

No behavioral indicators or methods of measuring *Emotional Regulation* in action were discussed in any work.

4.2.3 Adaptation Skills

Whereas skills in the Monitoring skillset focus on maintaining an awareness, skills in the **Adaptation** category characterize how a student responds to such awareness. For example, if a student notices that they are stuck, then they might seek out help from an outside source. Similarly, if a student recognizes they are not making progress as they expected, they might decide to go to the library to work more effectively.

Within Puustinen et al.'s taxonomy, these skills would be classified as forms of 'self-control'. This, along with 'self-observation' is part of the *performance* phase.

Help-Seeking

Student use of outside resources (especially course staff) to help troubleshoot is frequently recognized as an important cognitive control skill [3, 28, 39, 85, 125]. *Help-Seeking* requires students to observe and assess their specific situation in a number of ways. Doebling and Kazerouni note "Academic Help-Seeking involves…identifying that help is needed to surmount the problem, deciding whether and from whom to solicit help, and finally obtaining and processing help'' [28].

Deciding who to solicit help from was a concept discussed by a few studies within this corpus.

A consistent sub-theme among works that discussed *Help-Seeking* was to focus in on a student's approach to *Help-Seeking* by noting what kinds of social assistance they sought or what kinds of questions they asked. Falkner et al. [39] discussed friends as a valuable resource for help, while Lyon et al. [85] discussed getting help from TAs or instructors. The types of questions students asked was a point of discussion for Peteranetz et al. [109] who noted two different types of questions that students might ask. They write: "students asked questions

to both advance their own understanding (high-level) or find out what the instructor wanted (low-level)" [109]. These two aspects of *Help-Seeking* both appeared to consider a student's approach to seeking help once they decided help was needed.

Lyon et al. [85] also considered the timing of when a student asks for help within their study. Among the self-regulation strategies noted in their work, they included "allowing failure or to reach the point of failure before looking for additional help" [85]. Their study was the only one in our corpus to touch on calculating when to ask for help. However, other work within the field has noted how correctly gauging when to seek help can be a big factor in successful *Help-Seeking*. Marwan et al.broadly categorized unproductive *Help-Seeking* into two forms: help over-use (or abuse) and help avoidance [92]. Seeking help too quickly (meaning that they have not sufficiently tried solving the problem on their own) or too frequently can reduce how well help is retained or how much is learned. Conversely, never seeking help or working independently for too long can both be frustrating for the student and can impact how much time a student has available to complete the remainder of their work.

While many studies recognize this to be a key aspect of cognitive control, only a few studies have examined patterns in *Help-Seeking* behaviors. Ren et al.created a formalized "Design Recipe'' to help understand what happens in TA-run office hours. Students and TAs filled out a form after interacting to characterize what the student needed help with and what steps of the problem-solving process they addressed together. Students most frequently sought help with analyzing a task, translating function outlines into code, and testing code [125].

Doebling and Kazerouni focused on characterizing the *Help-Seeking* process more broadly. They found that students considered all resources useful, but tended to progress from informal and easily accessed forms of help to progressively more formal sources when they encountered a difficult problem. This commonly took the form of first using online resources, then asking peers, and only then seeking help from course staff if their question had

not been answered [28].

Doebling and Kazerouni also recognized how identity or looking foolish influenced help avoidance: "Computing students can be prone to comparing themselves to peers and taking actions to affirm their sense of belonging in computing.One such action might be help-avoidance in order to affirm one's identity as a computer scientist or programmer.'' [28]. This issue does not seem to be limited to *Help-Seeking* in CS courses either. For instance, Won et al.note that "...students may also associate seeking help with personal inadequacy and embarrassment and perceive it as a sign of weakness and a threat to their self-worth" [146]. Similarly Fong et al.note that those that avoid help experience less "emotional support and social efficacy" with instructors [41]. While this corpus of works did not delve deeply into this idea, these observations seem to indicate that identity is a critical variable to effective *Help-Seeking*. Further study examining a connection between sense of belonging and help avoidance could help us uncover a more nuanced view of when and why students seek help.

Reducing Distractions

Students who are skilled in *Reducing Distractions* are cognizant of how they study best and can assess and change environments in order to suit their needs. The MSLQ Time and Resource Management subscale includes several statements about a student making intentional choices about their study environment like: "I usually sit in a place where I can concentrate on my course" [114]. In a qualitative analysis of student self-reflection, Prather et al.reported Environmental Restructuring was a consistent theme [121]. Falkner et al.similarly reported "reducing distractions" among the themes from their qualitative analysis [39].

With only four passages discussing this skill, we could not identify any discernible trends or

specific behaviors that could allow an outside researcher to reasonably infer that a student is intentionally using this skill. For example, while knowing when a student has gone to the library to study could be trackable, this in itself does not demonstrate cognitive control without knowing intent. Instead, it is likely easier to observe this skill in a more simulated setting like a laboratory where a researcher could ask about a student's reasoning for choosing to study in a certain location.

However, *Reducing Distractions* is a key aspect of cognitive control within more general education literature. Ley and Young reported that 'environmental structuring' was highly related to effective Monitoring and noted this as the "strongest difference between college students who used an instructional self-monitoring protocol and those who did not" [77]. Additionally, besides being a significant theme of Pintrich's MSLQ inventory, it is also a key SRL strategy in the Zimmerman and Martinez-Ponz taxonomy [45, 166]. This theoretical foundation seems to indicate that *Reducing Distractions* could be a promising avenue for future study in CS classrooms.

4.2.4 Reflection Skills

The final phase of Puustinen et al.'s taxonomy is the appraisal phase wherein students consider their process and the work they produced in aggregate and resolve to make changes for the future [123].

With only nine total passages in this corpus discussing **Reflection**, this skillset receives much less attention than the previous two. Indeed, some of those nine passages provide little context, simply highlighting "self-evaluation" [121] or "Using Reflection" [85] as aspects of cognitive control.

One component of the Winne-Hadwin model is the "COPES" schema which details how

students evaluate success based on a set of standards [157]. While infrequent, the research team did notice two trends in what standards students applied when reflecting on past work. Sometimes researchers focused on how students reflected on assignment feedback [3] or, more generally, the expectations of the class [114]. These standards required *Reflection Using External Standards*, which come from sources external to the student. For example, the MSLQ includes the statement: "I try to change the way I study in order to fit the course requirements and instructor's teaching style" [114]. This is contrasted with *Reflection Using Internal Standards* which focused on how a student evaluated their work against their own, more internally facing, standards. For example, Stephenson et al. [144] report an intervention where students had to complete an exam wrapper that asked them to "reflect how they prepared for the exam, the kinds of errors that they made while answering the exam questions, and the changes that they would make for next time". In this situation, a student is reflecting on how they performed compared to their own expectations of themselves, rather than a class-set or instructor-set standard.

No behavioral indicators for reflection were noted within this corpus. Like *Reducing Distractions*, using just behavior to identify if a student is engaging with some **Reflection** skill cannot tell us much about their intent. Thus, making inferences between what a student does and what's going on in their head is difficult. Still, since reflecting is such a significant part of so many theoretical frameworks of cognitive control, any successful study to expose how a student goes about thinking back on their work could contribute to a better understanding of cognitive control as a whole.

4.2.5 Knowledge Building

Knowledge Building was not classified within any of the previous four skillsets as it could occur in any of Puustinen et al.'s three phases [123].

This skill encompasses the ways a student independently builds knowledge outside of what is needed for class credit. This skill was almost always discussed in the context of an indicator, rather than as a skill itself. Seeing students *practice* work on non-credit exercises [39, 110, 121, 162] was the most frequent indicator used. Most studies did not discuss this beyond noticing that practicing was important for success and an aspect of cognitive control. For example, Prather et al. [121] simply include "practice coding" as one of the self-regulatory skills important for CS success. Seeing students investigate, or "build knowledge by exploring other resources" as Falkner et al.put it [39], was also occasionally brought up as an indicator [109].

Similar to *Reducing Distractions*, *Knowledge Building* presents some difficulties for further study as it is difficult to measure intent. For example, an outside observer watches a student search for information on something outside of the course. This could be demonstrating investigation, or it could just as likely be demonstrating that a student has fundamentally misinterpreted what they need to be doing. Without the knowledge that a student is intentionally searching for information outside of what is required for a course, one cannot make an inference that the behavior indicates *Knowledge Building*.

4.2.6 Procrastination

Procrastination can be defined as intentionally delaying work in spite of an awareness of the negative consequences of the delay. Since it does not fit within our definition of skill, procrastination was categorized as a "general indicator" and is not within another one of our

skillsets. Within the works examined in this review, procrastination was regularly framed as a signifier of a difficulty regulating, rather than a cognitive-control skill [3, 130, 143, 162]. However, opinions differed on which specific skill difficulties were at the heart of the problem.

Procrastination is often thought of as difficulty with setting aside appropriate time. Arakawa et al.discuss procrastination as an underestimation of time in their analysis, noting some students habitually waited until just before a deadline while others realized that the time they set aside was insufficient once they were too close to the deadline [3]. Zhang et al.measured procrastination as the number of days before a deadline an assignment was started [162].

However, other papers in this corpus attributed procrastination to difficulty with other skills. Shaffer and Kazerouni [130] took a different approach and focused on both an awareness of time and an ability to decompose abstract projects. In their study, the authors introduced project milestones that were intermediate due dates for specific components of an overall project. These milestone helped create more tangible deadlines with straightforward work because "a task whose outcomes are farther in the future are more likely to invite procrastination". This intervention demonstrated *Decomposition* to students and provided a structured way to learn about *Monitoring Progress*.

Martin et al. [35] examined procrastination in other ways by examining interventions; each seeking to scaffold a different skill. Their first intervention focused on reflective skills and asked students to write a few sentences on their use of time in the prior assignment before starting the next project. Their second intervention had students develop their *Scheduling* ability by creating a written schedule sheet. Here, the aim was to help students "form, express, manage, and track smaller-scale deadlines" [35]. Third, automated email alerts detailing how far a student had progressed compared to their peers was introduced to help students *Monitor Progress*. While not explicitly conceived with this in mind, these three interventions focus on the three common phases of self regulation. Schedule sheets represent a

way to help reduce procrastination in the *preparatory* phase, automated emails help students self-observe in the *performance* phase, and written reflections train students to *appraise* their work.

Pereira and Díaz [107] recognize three different types of procrastination that are similar to the phases outlined by Puustinen et al.—though with different names. Before starting, students may have ''difficulty planning and prioritizing tasks“. While working, they may delay work because of an inability to focus and avoid distraction. After working, a student's choice to start working on similar tasks may be influenced by self-efficacy.

4.3 Discussion of RQ1: Coverage of Identified Skills

Our first research question was: *What aspects of cognitive-control skills have been examined in post-secondary computer science education research, and what behaviors have been used to indicate skills?* Thus, we seek to provide a summary of what areas within cognitive control have been targeted previously. This question also seeks to identify ways students demonstrate cognitive control. This is useful to know since, in order to assess an effective intervention, one needs to recognize when a student has improved. We answer our first research question through the data summarized in Tables 4.3 and 4.4. Our review identified 11 skills and 10 observable behaviors that can act as indicators of those skills in students. These skills were diverse in that they address what students do before, during, and after completing a task.

Areas of focus — While this list shows cognitive control work has examined a breadth of skills, there is a clear focus in this corpus on only a few of these. *Monitoring Correctness*, *Decomposing*, and *Scheduling* were the most discussed, while *Emotional Regulation*, *Reducing Distractions*, and **Reflection** were the least discussed. Passages highlighting *Emotional*

Regulation or *Reducing Distractions* came frequently from works using the MSLQ. However, outside of this general education inventory, they were not often discussed in the context of CS directly. This means the eight studies which used the MSLQ ended up being the vast majority of works to measure things like *Emotional Regulation* or *Reducing Distractions* [14, 15, 32, 75, 76, 79, 121, 139], albeit indirectly. Overall, there appear to be noticeably fewer studies examining how students respond and reflect as compared to how they plan and monitor. This observation appears to extend beyond the works in this corpus into other literature reviews on self regulation and metacognition [45, 135]. This is not necessarily a weakness of the current state of CER, as there is still so much we do not know about even the most well-studied skills. However, *diversifying which skills we examine could better inform instructors about what needs the most instruction in future.*

Help-Seeking and Procrastination — *Help-Seeking* and procrastination are two areas presenting special challenges, and these problems may need to be addressed in future study. Both terms describe complex and multivariate aspects of cognitive control, which makes comparing results of different studies especially difficult. As the field advances, the CER community would benefit from developing shared vocabulary that helps to tease apart the specific aspects of these broad constructs that are being studied in a particular work.

Help-Seeking encompasses a wide variety of judgements regarding whether one needs help, what question to ask, when to ask it, what resources to use, and what to do with the help once received [28]. These judgments are each qualitatively different from the others, and students could have different levels of proficiency at each. For example, a student could recognize the right time to ask a question, but struggle to form a question that will help them effectively learn. Similarly, procrastination can indicate a failure before, during, or after working on an assignment [162]. This makes it difficult to compare the results of different interventions for these areas since researchers are typically intervening only on

specific components of a whole. Shaffer and Kazerouni [130] created an intervention seeking to reduce procrastination by creating milestones that forced students to decompose, while Pereira and Díaz examined the efficacy of automated reminders so as to scaffold *Monitoring Progress* [107]. These studies are all seeking to reduce procrastination. Yet there are so many different aspects of this issue to intervene on. A student might quickly Decompose a task, but end up procrastinating when they do not stay aware of how much time they are spending on a part of the problem. Alternately, a student might have a strong track record of managing time effectively but then poorly allocate their time due to difficulties with *Decomposing* the task. Thus, all of these studies seek to intervene on procrastination, but approach the problem in such different ways that it is difficult to compare Shaffer and Kazerouni's work to that of Pereira and Díaz. Thus, future work may benefit from attempting to break *Help-Seeking* and procrastination into smaller components.

4.4 Relating Skills to Success

In this section, we discuss the degree to which skills studied in this corpus have a strong or promising connection to academic success.

Table 4.6 provides an overview of the 24 studies that related a skill to academic success in some way. We use the word 'observe' to mean that a study noted a relationship between a skill and success, but did not present any statistical analysis. This means results from qualitative studies will be reported as observing a relationship between a skill and success. Additionally, there are several studies that reported differences in average or median score between control and experimental populations, but did not test to see if the difference is statically significant. These studies were also categorized as reporting an 'observed' relationship. This is not to say these observed relationships are not valuable. Indeed, these studies help point towards

promising new directions for future work. Instead, we use the word observed in this way to communicate a difference in the generalizability of the results found.

Skill	Statistically Significant Relationship With Success	No Significant Relationship With Success	Observed Relationship With Success	Unique Studies Relating To Success
Planning (skillset)		[136]		1
Task Analysis (skill)	[25, 81]		[117, 118]	4
Scheduling (skill)	[18, 26, 62, 74, 162]		[3, 74, 85]	7
Decomposing (skill)	[81, 130, 145]			3
Monitoring (skillset)				0
Monitoring Correctness (skill)	[24, 73, 81, 145]		[3, 117, 132, 143, 151]	9
Monitoring Progress (skill)	[81, 145]		[117]	3
Emotional Regulation (skill)				0
Adaptation (skillset)			[18]	1
Help-Seeking (skill)			[85]	1
Reducing Distractions (skill)				0
Reflection (skillset)		[85]		1
Reflection - Reflection Using Internal Standards (skill)				0
Reflection - Reflection Using External Standards (skill)		[144]		1
Knowledge Building (skill)		[162]		1

Table 4.6: Studies Which Relate a Skill with Academic Success Metrics (Does Not Include Papers Using the MSLQ)

MSLQ As described in Section 4.2, the various forms of the MSLQ inventory often aggregate several skills together to form one holistic score that describes a student's proficiency with self regulation. Therefore, Table 4.6 does not report on any studies that used this inventory to measure cognitive control. Instead, Table 4.7 lists each study, the version and subscales of the inventory they used to measure self regulation, and whether or not the scores shared a statically significant relationship with academic success.

Only two studies reported on positive correlations between MSLQ scores and academic success metrics. Duvall et al. [32] found a moderate positive correlation between four individual statements from the MSR subscale and concept inventory scores, but do not note whether the relationship is significant. Prather et al. [121] reported a weak but significant positive correlation between success metrics (like test score and project score) with the sum of three

MSLQ questions from the MSR subscale.

Reference	MSLQ Version	Subscales Used	Statistically Significant Relationship With Success
[32]	MSLQ-1991	Metacognitive Self-Regulation Effort Regulation	Unreported
[15]	MSLQ -Columbia	Planning Monitoring Study Method Regulation	No
[121]	MSLQ-1991	Metacognitive Self-Regulation (3 questions)	Yes
[14]	MSLQ-1990	Metacognitive Self-Regulation	No
[79]	MSLQ-1991	Metacognitive Self-Regulation	No
[139]	MSLQ-1990	Metacognitive Self-Regulation	No

Table 4.7: Results relating MSLQ Inventory Scores to Academic Success

4.4.1 Skills With a Demonstrated Relationship to Success

Monitoring Correctness

As the most frequently discussed skill in this corpus, it is perhaps no surprise that *Monitoring Correctness* bears a consistent relationship with success.

Predicting Confidence is measured in several works within this corpus. Predicting Confidence involves asking students to report on their confidence in their answers immediately after they finish an exam. Stephens-Martinez writes: "Generally, our students are reasonably accurate within 10% of their actual grade", and that observation seems to be consistent with papers both within the corpus [24, 132, 143, 151] and more generally as well [55]. Most studies also report that lower-performing students are worse at these predictions than middle or high performers [24, 73, 143], and they also tend to be overconfident.

While these predictions are a useful tool for measuring a student's proficiency with Monitoring Correctness, this approach does come with some limitations. Denny et al. [24] and Stephens-Martinez [143] both note test predictions did not significantly change in accuracy throughout the course of a semester, implying they are not a good catalyst for behavioral change. Additionally, while they do appear to be an effective tool for measuring *Monitoring Correctness*, there could potentially be a ceiling to their efficacy for even this purpose. If students are already fairly accurate at predicting their confidence, then an intervention seeking to improve their proficiency with Monitoring Correctness can do little to increase how accurate they are at the specific behavior of Predicting Confidence.

Validating is another approach to observing *Monitoring Correctness*, though evidence for this indicator's relationship to success is more conflicted. In their Think-Aloud study, Sudol-DeLyser [145] examined how self-explanation at different levels of abstraction differed among higher and lower performers. They found no significant differences between their success metric (number of submissions to an AAT) and different types of self-explanation. However, they did find a significant difference in the types of self-explanation that High Performers engaged in versus Low Performers. High performers more frequently engaged in self-explanation that related disparate elements together or to the overall algorithm. They note: "This would indicate that higher proficiency students are more likely to use abstract statements during the self-explanation process" [145]. In Loksa and Ko's Think-Aloud study, 'self-explanation' was a code used to describe when students verbalized "an account of why a decision was correct" [81]. In their multiple linear regression of CS2 students, increased frequency of these statements significantly corresponded to an *increase* in errors in code. That more self-explaining correlated with more errors seems to indicate the behavior is more frequent in students who are struggling. This appears to conflict with Sudol-DeLyser's findings, but Sudol-DeLyser used submissions to an AAT as a success metric while Loksa and

Ko evaluated code by looking for errors.

As *Monitoring Correctness* seems to already have a clear relationship with success, we would expect validating behaviors that demonstrate that awareness to also have a close relationship with success. However, as only these two studies examined validating behaviors in this way, we can only note further examination into this behavior is likely needed.

In Loksa and Ko's Think-Aloud study [81], the authors had an opportunity to be in the room as students verbalized their process of *Monitoring Correctness*. Statements where students reflected about their "the understanding of code or problem prompts" [81] were coded as Comprehension Monitoring. Loksa and Ko's 'Comprehension Monitoring' category combines what we have defined as *Monitoring Correctness* and *Task Analysis*. In their linear regression of CS2 students, the frequency of this code significantly related to a decrease in code errors. Since this code aggregates two different skills from this review, it is unclear how significant this finding is when relating *Monitoring Correctness* to success.

Overall, it appears that there is a correlation between higher performing students and an awareness of correctness, but this corpus largely focused on one method that does not effectively scaffold improving the skill of *Monitoring Correctness*.

Scheduling

Several studies relate indicators of *Scheduling* to success. Three studies showed a significant correlation between starting early and academic success, one showed a statically significant relationship between spacing and success, and one used both indicators together in an intervention that particularly helped weaker students.

Starting early was significantly related to academic success in three papers from the corpus [26, 74, 162]. Denny et al. [26] used optional early feedback deadlines as an incentive

to help encourage starting early. They found students who engaged with early deadlines earned significantly better grades compared to those who did not. However, the degree to which starting early is important is not yet clear. Zhang et al.[162] saw weak but significant correlations between starting early and homework grades overall. They also found a "weak but statistically significant correlation between average DSBD [days started before due] over the homeworks completed before the first exam'' and that first exam score [162]. However, they could not replicate these findings with starting homework early before the second exam and that exam's grade. Leinonen et al.noted a "clear trend" where students who started earlier on problem sets outperformed those who did not, but also note "even of those who start one day before the deadline, most will get a 5, i.e.the best grade available.Only for those who start on the very last day, the day of the deadline, the most likely grade is 0 or 1" [74]. For these homework assignments, the key was not how early a student started, but that they left at least a day to complete the work. Thus, it may be valuable for future work to investigate how early is early enough for different types of assignments.

Overall, these studies show evidence that starting early has a significant correlation with academic success. However, the degree to which it is important is still not yet clear.

Spacing refers to the process of distributing smaller time blocks throughout the course of an assignment. In this corpus, only one paper related this skill to academic success, but the relationship they found appears promising. Spacing is in contrast to 'cramming' (allocating one large block for all work) [75], typically right before a due date. Chung and Hsiao [18] studied the consistency of students using an optional practice platform. Of the students who interacted with the system, those who consistently used it week-to-week outperformed those who used it only right before an exam or inconsistently. However, lower-performing students who regularly engaged with the platform did find value in this pattern. When examining the behaviors of just low-performing students who consistently used the platform, they note

students who transitioned to more cramming patterns outperformed those who did not.

Starting early and spacing were used together in one study [62]. Ilves et al.gathered data on starting early and spacing behaviors compared to the class average. This was presented to students in two different ways: textually and through a graphic visualization. They noted among the high performing students that the interventions made little difference, but among the lowest third of the class, the graphical presentation had a statically significant relationship with exercise performance compared to the control [62]. In this intervention, we cannot determine whether one indicator contributed more than the other, but increasing overall awareness seems to have helped.

4.4.2 Skills with a Promising Relationship to Success

Skills discussed in this section have some evidence that they hold a relationship with success, but more study is needed before we can say for sure.

Task Analysis

Task Analysis has been observed to be important, but statistically significant results have not replicated these findings. Four studies sought to connect *Task Analysis* with academic success [25, 81, 117, 118].

As mentioned in Section 4.4.1, Loksa and Ko's Comprehension Monitoring code encompassed both *Task Analysis* and *Monitoring Correctness*. This finding, therefore, indicates a statically significant relationship between academic success and this pair of skills. However, it is unclear how direct this relationship is.

In 2018, Prather et al.performed a Think-Aloud study looking for common metacognitive

difficulties that students face when using an Automated Assessment Tool (AAT). They observed that proficiency at forming a correct conceptual model was one of the biggest differences between students who successfully completed the assignment and those who did not. The authors summarize: "many of the students who did not complete the quiz read the prompt (often briefly) and jumped directly to coding...This proved disastrous for them as they wandered aimlessly, seeming to hope they would eventually stumble on a solution" [117]. This study focused on characterizing metacognitive difficulties that students encountered when interacting with an AAT, and part of the conclusion was that these tools need to provide "implicit support of metacognitive awareness" [117]. One area they suggest for future work is to create software that indicates whether a student understands what they are doing before beginning to code.

Prather et al.and Denny et al.both sought to create interventions to do just that. All students were given a programming problem but students in the experimental group had to first "solve a randomly generated test case after reading the problem prompt" [118] before they could begin coding. The major differences between these two studies were the methodologies and population sizes used. Prather et al.conducted a Think-Aloud study on a small population (n=38) where all students were observed interacting with the programming problem and intervention. They observed "the experimental group had a higher completion rate, faster time, and fewer attempts required to complete [the assignment]" [118].

Denny et al. [25] conducted a larger scale (n=976) quantitative follow-up study where students in the experimental branch were required to solve a test case before starting a homework assignment. They reported that students in the experimental group had a statically significant reduction in logic errors compared to the control. However there was no significant difference in completion time, success rate, or number of submissions to the AAT. These findings seem to largely contradict what was observed by the previous two Think-Aloud

studies, but Denny et al.do note several limitations in their study and recognize the need to replicate the experiment "within a more structured laboratory environment" [25] to be more confident in their results.

Prather et al.'s 2018 study [117] demonstrated the importance of *Task Analysis* for students in CS, noting that jumping into coding before starting "proved disastrous" for students who did not fully read the prompt. The subsequent two studies then tried to analyze the effect of preventing this specific behavior by not allowing students to code until they had solved a test case, but these had mixed results. At a minimum, a third follow-up study would be needed to validate whether preventing students from jumping into coding helps scaffold this skill. There is also further work to be done in this area to assess what kinds of assignments this intervention works for. Is there value in such test cases for coding assignments which are larger in scale and complexity? What about more theoretical assignments?

There is also more work to be done assessing other facets of this skill. As stated in Section 4.2.1, correct understanding of the task is desirable, but *Task Analysis* requires that students know how to intentionally take time to internalize requirements before starting. Asking students to solve test cases before starting work is one approach to encouraging this behavior, but it not the only way to do this. The efficacy of encouraging students to annotate the problem statement, for example, could be another important avenue for future investigation into this skill.

Decomposing

While *Decomposition* is one of the most frequently discussed skills in this corpus, it has seen limited study relating it to academic success. Out of the eighteen studies that discuss this skill, only two relate *Decomposition* to success. However, both of these studies found

promising evidence. In their Think-Aloud study [81], Loksa and Ko's regression model relates the frequency of their categories among students to errors in the code they wrote. They found that frequency of statements related to starting or revisiting sub-goals significantly related to fewer errors among more experienced students. Shaffer and Kazerouni created an intervention where students had to meet mandatory intermediate deadlines where different sub-goals of a longer-term project would be due. While they sought to reduce procrastination on programming projects, their implementation focused on delays "stemming from difficulties with decomposing a large engineering task" [130]. They report that while pass, fail, and withdraw rates stayed the same, students in the intervention group overall produced code that was significantly more correct than students in the control. As a result, the grade distribution changed with the effect of adding a letter grade to the middle third of the class.

Given these results and the fact that Decomposition is so widely acknowledged to be an important part of cognitive control, this skill holds great promise for future research. It is especially notable that none of the commonly discussed indicators for this skill in Section 4.2.1 (enumerating and outlining) were studied in relation to academic success in any work in this corpus.

4.5 Discussion of RQ2: Common Patterns among Studies

Our second research question was: *Which cognitive-control skills (or associated behaviors) have a demonstrated relationship with post-secondary student success in computer science education?* In this section, we highlight three major takeaways from the analysis of Section 4.4. First, it is likely that study of the field of cognitive control would benefit from greater di-

versity in what skills we study and how we approach that study. Second, cognitive-control interventions appear to impact students at different performance levels differently. Third, the MSLQ may need some additional validation within a CS classroom as it was a weak predictor of success within this corpus.

1.Diversifying Study of Cognitive Control: As discussed in Section 4.3, diversifying what we study could be beneficial. As Table 4.6 shows, there are 24 studies that relate any skill identified in this work to success. 16 of those studies (66.66%) relate *Monitoring Correctness* or *Scheduling* with success. It is possible that other aspects of cognitive control, such as **Reflection**, are just as valuable for CS students. Without further examination we cannot say for sure. Thus, future research into what skills most directly impact academic success could help educators pinpoint what parts of cognitive control to intervene on.

Additionally, future work might find it valuable to diversify the methods we use to examine cognitive-control skills. For example, there are five studies in this corpus that examine *Monitoring Correctness* by asking students to Predict Confidence. *Monitoring Correctness* overall is widely agreed to be valuable, but confidence predictions do not appear to be a causative skill that improves performance. Additionally, the context of these confidence predictions was limited to exams. Extending these types of questions to other types of assignments, like programming homeworks or larger-scale projects could be a valuable new space for this form of measurement. Predicting Confidence is just one example of the need for diversity. Scheduling was most often measured by noting when students started assignments. Studies examining *Task Analysis* focused mostly on discouraging rushing.

Along with different methods, it might be helpful to broaden the indicators used to capture cognitive control. There was a trend among studies in this corpus to use indicators that exemplify how educators *want* students to act. This is not to say such indicators are unhelpful

for research, just that they cannot describe cognitive control in isolation. For example, it is easy to make the inference that a student who Starts Early or Spaces out their work is intentionally *Scheduling*. Conversely, the student who started one day before the due date might have intentionally allocated the right amount of time, or could be scrambling to finish. As outside observers, we do not have the means of differentiating from a timestamp alone. However, a student who allocates the last six hours before a deadline to work because they assessed their work and Scheduled accordingly is still demonstrating cognitive control. Starting early and spacing also do not take the greater context of a student's situation into account. For example, *Scheduling* behaviors might look different for a student with many other commitments (like a job or a heavy course load) than it does for others.

2.Cognitive Control for Classroom Equity: Within this corpus, it became clear that cognitive control interventions might be more equitable than equal. There were several studies that found interventions helped students who were struggling to succeed more dramatically than students who were already succeeding. This observation makes intuitive sense as those that have stronger cognitive control skills are more likely to succeed in any context. We highlight it here, though, as students who are struggling tended to respond to cognitive-control interventions in qualitatively and quantitatively different ways that might not be visible when evaluating a class as a whole.

Ilves et al. [62] sought to scaffold *Scheduling* by creating an intervention where students in the experimental branch saw visualizations of how their work patterns differed from class averages. When comparing overall populations, they reported no significant differences between control and experimental populations. However, when they stratified their students into thirds based on performance, it was clear that students in the top tertile tended to do well with or without an intervention while students in the lowest tertile were significantly

more successful with an intervention. Other studies have also noticed that students at different performance levels respond differently to cognitive-control intervention. Denny et al. [26] noted that struggling students saw more dramatic improvement than their high-performing peers. Shaffer and Kazerouni found that their intervention helped students in the middle more dramatically than high-performing or low-performing students [130]. Thus, future work may find it valuable to measure the impact of a cognitive-control intervention on different brackets of student performance as well as the impact on the class overall.

Chung and Hsiao [18] also observed a situation where students demonstrated cognitive control in a qualitatively different way. In this study, the authors examined how different patterns of engagement with an optional practice platform related to academic success. When looking at the class overall, Chung and Hsiao reported a moderately positive correlation between consistent engagement and exam score. However, they also examined transitions in engagement patterns among different populations of students, including students who failed the first exam of the class who consistently practiced. Among this specific population, those that moved from a consistent pattern of engagement to a cramming pattern (practicing just before an exam) outperformed their peers in subsequent exam scores. The authors hypothesize that this bracket of lower-performing students, those that engage consistently "...do not practice effectively...having a high error rate or gaming the system for memorizing more questions and answers" [18]. In this case, switching to a cramming approach might have helped these students take a more thoughtful approach. We highlight Chung and Hsiao's results here as it shows a way in which the successful approach for the class overall and the successful approach for lower-performing students did not align. This was the only study to report such a finding within our corpus, but examining how different performance levels of students demonstrate cognitive control could be an interesting avenue for future work to explore.

3.The MSLQ and Academic Success: MSLQ scores do not seem to reliably predict success among works in this corpus. While certain statements from the inventory have been discussed in previous sections, it is important to note that most studies in this corpus take the aggregate score of all of those questions. No significant correlations were found between aggregate scores and success [14, 15, 139], nor did any study conducting a regression analysis note the MSLQ as a significant factor [14, 139]. Lishinski et al.conducted a path analysis, but they also did not see a significant relationship between MSLQ scores and any success metric [79]. While there is a little evidence that aggregating a smaller number of questions correlates with success [32, 121], there are a few pieces of evidence that indicate this inventory may not be useful for modern CS students. Within the study of education generally, Broadbent and Poon noted that the MSLQ: "...may not capture the construct of online learner self-regulation as accurately as online-focused, validated measures" [11]. This makes some sense since the most recent inventory was published in 1991—meaning it predates the adoption of Google as a search engine by a decade. Thus it is possible this inventory is not asking questions that accurately measure the study strategies of modern CS students. Within CS education, Prather et al. [119] have also critiqued self-reporting as a method of measuring cognitive control. They note that "self-report measurements of cognitive control, such as the MSLQ, often measure what students think they do, rather than what they actually do" [119].

4.6 Recommendations for Future Research

Overall, future work may benefit from diversifying both what and how we approach the study of cognitive control. Some skills, like *Reducing Distractions* or *Emotional Regulation*, have firm theoretical backing but were rarely the focus of studies in this corpus. Other skills,

like *Task Analysis* and *Decomposition*, appear promising, but need more evidence to conclusively relate them to success. Even frequently studied skills, like *Monitoring Correctness* or *Scheduling*, could benefit from novel approaches to better understand how they work.

From conducting this review, it is clear that even among the most frequently studied aspects of cognitive control, there is still much to explore. It is our hope that this review will inform and spur future work so that students will better learn how to effectively learn. The works in this review identified a wide variety of skills a student will use before, during, and after working. Still, finding conclusive evidence that many of these skills related to success in a meaningful way was not possible. In many cases skills, such as *Monitoring Progress*, have simply not seen enough focused study within this corpus to identify trends.

We echo recommendations by Prather et al.and Loksa et al. [84, 119] that precise language to discuss what we as a community mean when we talk about cognitive control can only help us better understand this construct. We also need greater precision in our language about individual constructs. For *Help-Seeking*, procrastination, or even the term 'planning', it may be helpful to come to some consensus about what precisely these words mean and what is within their scope. Getting more precise about these terms could also help us mentally re-frame these aspects of student learning to and identify better ways to support student learning.

More data points can help the research community better come to know this aspect of the student learning experience. We hope that within this review, we have offered some promising directions for that future research. From the skills we study, to the behavioral indicators we look for, to the very ways we construct our studies, there appears to be ample opportunity for future work to build off of the foundations of existing work. Novel approaches or frameworks for understanding cognitive control may not lead anywhere themselves, but could help clarify what does and does not work about our existing ideas on this subject.

Overall, it appears that the study of cognitive control currently offers a great deal of space for creativity and new ideas in studying student learning within CS. The field of Learning Analytics is starting to be leveraged to study these skills unobtrusively and even help students learn how to be more aware of their own habits. Thus, with a lot of reasons to try something new and a lot of new tools to try, it appears that it is a particularly exciting time to study cognitive control.

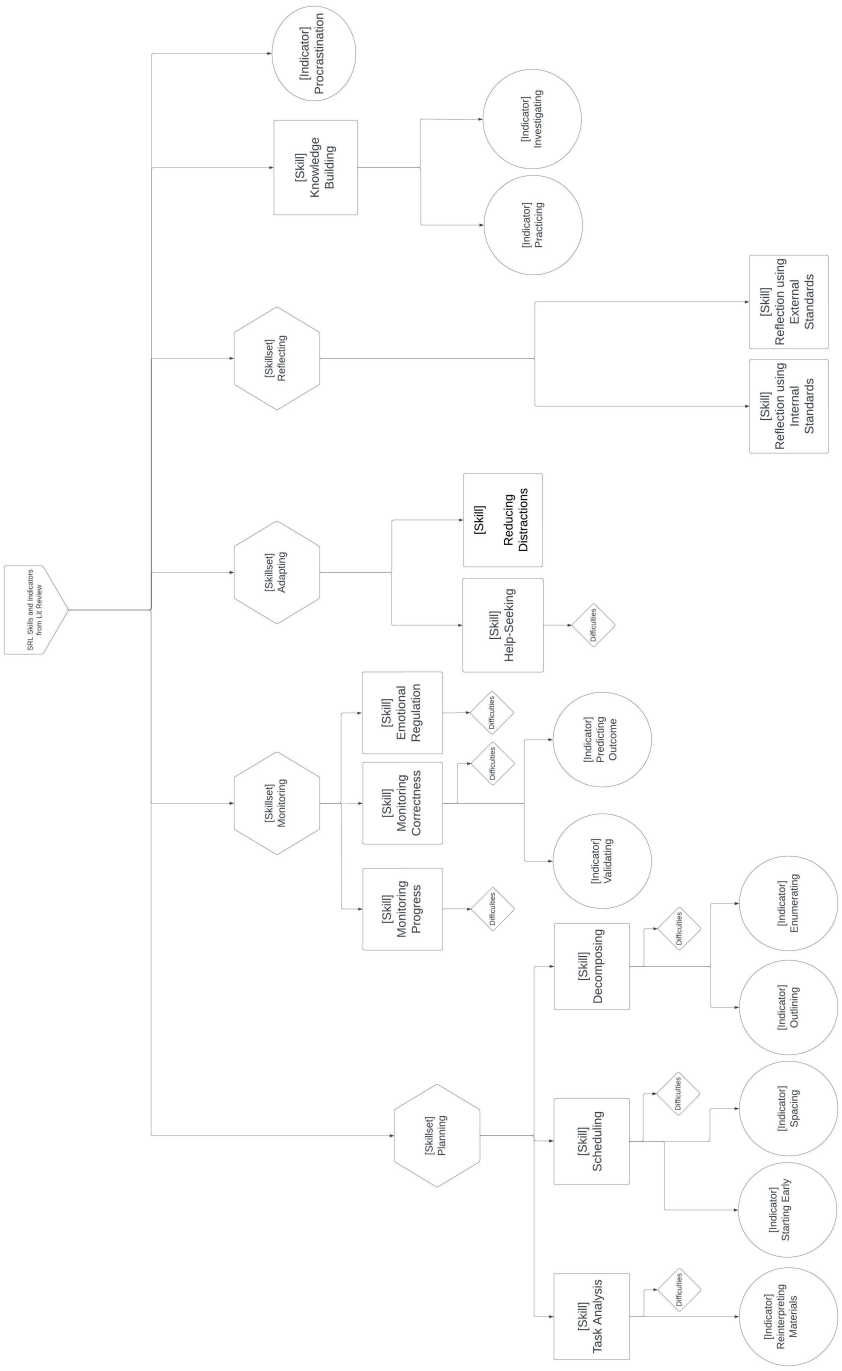

Figure 4.3: Hierarchical description of all identified skills and their skillsets

Chapter 5

Identifying Data Traces

This chapter is text that is pulled largely from 'Taking a Step Back: Assessing the Feasibility of Using a Wider Digital Ecosystem to Study Self-Regulated Learning Skills' (to be submitted to, 'Frontiers in Education'). This means the term 'we' is used. The authors on this study were (in order): Molly Domino and Clifford A.Shaffer. Thus, 'we' refers to that list of authors.

In the previous two chapters, we focused on identifying skills. The Delphi Process, identified skills that CS educators found important. The literature review, identified skills that the CER community have focused on. Those studies provide a list of skills that have strong support.

We now turn to considering how to measure those skills. This is a critical next step for two reasons. First, as with any intervention-based research, there needs to be some way to assess students proficiency with a given skill before the intervention begins and after it finishes. Otherwise, it would be impossible to determine if the intervention had an impact on skill proficiency. Second, to improve a skill, students will need to go through iterative cycles of practicing using the skill and receiving feedback. In order to accomplish both of these things, researchers and educators need a way to measure those self-regulation skills.

How, then, can anyone watch a highly internal skill develop? Within the context of learning to program, this question is especially significant since the major activities are often done by

an individual in private — making study through direct observation impossible. We focus on identifying data traces to observe self regulation because they offer the opportunity to capture authentic learning empirically and at scale in a way that Think-Aloud studies and self report inventories cannot accomplish (see Section 2.2 for more detail).

Within this chapter, we first revisit and refine our list of skills as the Delphi Process study and systematic review leave us with some discrepancies. Second, we define what a 'good' data trace is and detail our process for identifying one. Third, we assess the software tools common to the digital ecosystem of most CS classes to see which traces we may already be tracking. Fourth, we present our finalized list of skills, data traces, and sources for those data traces. This is meant to help researchers consider broader data sources for future work. Finally, we return to Help-Seeking. As discussed in Section 4.3, Help-Seeking is a particularly difficult skill to study since it is particularly complex and hard to directly observe authentically [28, 125]. We present a state diagram of the various components of help-seeking and note how those phases could be observed through traces.

In this chapter, we also make the case that it is possible to take a wider perspective with our data collection efforts. By 'wider perspective', we mean that, rather than having a single data trace provide an inference about a student's proficiency with self regulation, we use multiple data traces from a variety of sources to evaluate a skill. Currently, most research using data traces tends to gather those traces from a single source (typically an LMS). However, from evaluating the digital ecosystem used in CS classes, it is clear that it is already possible to gather traces from a broader variety of sources. As noted at the end of the previous chapter, more data could help us gain a better understanding of how students regulate their learning.

5.1 A Finalized List of Self-Regulation Skills

The two studies detailed in Chapters 3 and 4 yielded somewhat different lists of SRL skills. CS educators identified 14 skills from the Delphi Process, while the systematic review yielded 11 skills. Overall, 15 unique skills were identified across both studies. While there was overlap on many of those skills, there were some discrepancies and changes that needed to be resolved first. This section goes over how those discrepancies were resolved to end up at the finalized list of 12 skills shown in Table 5.1.

First, Working with Peers and Group Mates was a skill identified in the Delphi Process, but not in the literature review. During the Delphi Process, there was some lingering disagreement as to whether it was a self-regulation skill or a co-regulation skill. At the time of the Delphi Process study, we believed that it fell on the side of self regulation (though the definitions did offer some ambiguity). However, combined with the lack of evidence from the systematic review, it seems prudent to hold off on studying this skill further until the research community better understands it. Additionally, our goal is to identify data traces that let us observe self-regulation in action. This skill is likely a poor candidate for future trace-based study. The question of how to attribute a data trace from a single computer to specifically group-work likely requires devoted study all of its own.

Two other skills, Adaptation and Starting Early, are determined to be too wide and too narrow in scope, respectively. Adaptation represents a group of skills, qualifying as a skillset under the systematic review. Conversely, Starting Early seems to be more of an indicator of Scheduling in the literature review than a skill in its own right.

However, we chose not to adjust all skills from the Delphi Process to match the the findings of the systematic review. Within our systematic review, the two forms of help-seeking were merged together. I opt to keep these as separate skills as they appeared to have distinct

Skillset	Skill	Finalized Definition
Planning	Task Analysis	Forming an accurate conceptual model of the task at hand
	Scheduling	Intentionally allotting blocks of time in the future to complete the task
	Decomposing	Taking an abstract task and breaking it into smaller, more concrete sub-tasks as a way to construct an overall strategy or algorithm
Monitoring	Monitoring Correctness	Maintaining an awareness that work is being completed correctly
	Monitoring Progress	Maintaining an awareness of time and pace of work while completing a task
	Emotional Awareness	Maintaining an awareness of current emotional state and potential impact those emotions could have on work
Control	Knowing How to Seek Help	Assessing what questions to ask and what channels are most useful for help in their situation
	Knowing When to Seek Help	Assessing if asking for help would contribute more to learning the material than working independently
	Reducing Distractions	Making an effort to adjust study environment in order to maintain focus
Reflection	Reflection Using Internal Standards	Assessing their work in the context of their expectations of themselves
	Reflection Using External Standards	Assessing their work in the context of course expectations
Phase-Independent	Exploration	Practicing outside of the scope of the class

Table 5.1: Finalized List of Self-Regulation Skills Important for Success in CS

behavioral indicators and therefore likely had distinct data traces.

Procrastination, which frequently appears in the literature, is also removed from our finalized list. This is because it is not a skill but a ‘generalized indicator‘, and specifically indicates some sort of difficulty with SRL. However, as explained in Section 4.2.6, researchers have not yet reached a consensus about what precise SRL skill it indicates difficulty with. While it is likely possible to identify procrastination with unobtrusive data traces, it could not lead back to a specific SRL skill. Thus, as this chapter focuses on measuring skills, this was removed.

Removing those two brought our list of fourteen skills down to 12, as outlined in Table 5.1.

Skill	Operations	Products
Task Analysis	Inferring requirements that were not clearly stated in the instructions	Work contained inferred requirements
		Record of student seeking clarification
	Forming a correct conceptual model of the task	First draft of work demonstrated a correct model
		Solving test cases before beginning
		Having a first submission that matches requirements
	Reinterpreting materials to make sense of the task	Took notes on instructions
		Highlighted important components of instructions
	Reading assignment thoroughly	Spent active time with assignment open (before getting started with work)
	Searching for relevant information	Accessed where that relevant information is stored
	Identifying unclear instructions and seeks additional information from instructors	Written record of student seeking clarification
Scheduling	Allocating time in a deliberate manner	Wrote plan for time allocation
		Used a tool (like a planner or calendar)
	Checking in on due dates before beginning	Accessed information
Decomposing	Articulating a set of sub-tasks	Outline, to do list, or other sketch of work to be done
	Focusing in on a single sub-task	Different drafts of work focus on specific goals
	Student applied some form of prioritization to their list of sub-tasks (they picked some item to start first)	Student works on one feature of the assignment
Monitoring Correctness	Validating work completed before continuing	After completing some unit of work, that unit is edited and evaluated before progressing
		Rubric (or other assessment tool) open while working
	Assessing correctness on individual test/assessment questions	Removed as it requires a self-report to measure
Monitoring Progress	Student seeks extension	Student communicates need for extension with course staff
	Student compares progress to their own expectations	Removed as it requires a self-report to know a student is performing comparison (obtrusive)
	Student compares progress to due date	Removed as it requires a self-report to know a student is performing comparison (obtrusive)
	Student compares their progress to the time they have allocated	Removed as it requires a self-report to know a student is performing comparison (obtrusive)
Emotional Awareness	Demonstrating patience and an internal locus of control	Removed as it requires a self-report to know if a student is indeed regulating this emotion (obtrusive)
	Demonstrating resilience towards failure	Removed as it requires a self-report to know if a student is indeed regulating this emotion (obtrusive)
	Taking a deep breath	Removed as it requires a self-report to know if a student is indeed regulating this emotion (obtrusive)
Knowing How To Seek Help	Asks specific questions regarding work	Written documentation of questions asked
	Utilizes a variety of resources to get answers	Records of access to that variety of resources
	Coming up with a hypothesis of what is wrong	Removed as it requires a self-report to know if a hypothesis was made (obtrusive)
	Using predefined strategies to methodically help them find the point of confusion	Removed as it requires a self-report to know if a student is intentionally doing this (obtrusive)
Knowing When To Seek Help	Tries some number of strategies before seeking help	Removed as it requires a self-report to know if a student is intentionally doing this (obtrusive)
		Spent time working before seeking help
	Seeks social help when needed, but not constantly	Frequency of office hour attendance

Table 5.2: Operations and Products for Skills (Table 1 of 2)

Skill	Operations	Products
Reducing Distractions	Starting a learning session in one location and moving to another location	Removed as it requires a self-report to know why a student changed locations (obtrusive)
	Assesses the qualities of a desirable location for work (possibly can articulate them)	Removed as it requires a self-report to know that a student was assessing or the results of that assessment (obtrusive)
Reflection Against Internal Standards	Students able to articulate their standards	Removed as it requires a self-report for students to articulate standards (obtrusive)
Reflection Against External Standards	Student revisits rubric	Student reviews rubric after working
	Internalizes feedback from returned work	Makes changes on an assignment after viewing previous assignment feedback
	Clarifies unclear feedback	Record of student seeking clarification
	Picking up or reviewing feedback	Engaging with an exam after it is graded
		Engaging with an assignment after it is graded
Exploration	Investigating	Student asks tangential questions
		Tinkering
	Practicing	Engaging with optional practice problems
	Practicing outside of the scope of the class	Engaged with practice items

Table 5.3: Operations and Products for Skills (Table 2 of 2)

5.2 Approach to Deriving Traces

At time of writing, there appear to be no verified or regularly used frameworks methods for selecting quality data traces. While there are frameworks for creating LA platforms more generally [91, 93, 134], none of these frameworks offer much actionable guidance on how to select effective data traces to study—focusing more on how work is analyzed. Therefore, this chapter summarizes the criteria used to evaluate traces and our rationale for our approach.

5.2.1 What Makes for a Good Data Trace?

Within this work, I evaluate data traces collected by software tools common to CS classes using the following criteria: validity, reliability, and equitability.

1. **Validity:** Perhaps the most intuitive of the three criteria, validity considers how

effectively the data gathered allows researchers to make an inference to an internal construct. Gray and Bergner ask: "do questionnaire answers or facial expression, actually measure *boredom*?" [48] as an example of how one questions the validity of the inference between a metric and an internal state. Going further than this, Winne et al. note that traces should provide "an objective (i.e., readily agreed to) account about how a learner operates on particular information at a point in time and in a relatively well-identified context" [160]. Without doubt, the best way to assess validity would be to validate the traces here against some ground truth (a challenge we discuss in Section 6.1). For now, we largely assess if a trace from a particular source lends itself to a clear inference, and whether the nature of the source could potentially harm validity.

2. **Reliability:** Gray and Bergner define reliability as 'repeatability or consistency of the instrument observations' [48]. They go on to explain that this criterion is similar to evaluating for random error, or how effectively a measure will capture the same thing in different contexts. To some degree, this comes down to how we select and standardize our metrics and thresholds. How bored does a facial expression need to look to qualify as 'very bored'? How many times should a student's code fail to compile before we classify them as being 'stuck'? These sort of questions are matters of reliability within measures.

   However, with quantitative measures, the bigger issue becomes one of portability, or how this metric transfers to other class contexts. How effectively does a metric taken in one class generalize to another class context? Portability is difficult to achieve, as courses will likely use the digital tools discussed in this paper in qualitatively different ways. Two classes could use the same technology in the same way, but if it represents a greater part of a final grade in one class, it could see different patterns of use. Still,

portable data traces are highly desirable as they can be meaningfully used by different teams of researchers in different contexts [20] and have the opportunity to be more widely validated.

3. **Equitability:** Equitability in a system means that the data is captured from all levels of students, from those that are strong at SRL to those who struggle to master these skills. For example, the only students who will ever engage with an optional tool (like ungraded practice platforms) are those who are aware of the tool and find it valuable to their learning. In essence, such optional systems require a baseline level of self regulation and motivation to use. This leads to greater reporting bias from students who already exhibit strong SRL skills. Conversely, null data does not communicate much meaningful information about the students who are still learning to self-regulate effectively.

   Equitability is important to consider given the results of my systematic review (see Section 4.5). Within that study, it became clear that students who are struggling to self-regulate behave measurably differently from students who are already adept at SRL and, therefore, measurements from one population do not generalize to the other.

Matters of Scope

LA literature identifies data traces for SRL at two different levels of scale for inferences: macro- and micro-level traces. Macro-level data traces are any traces that identify self-regulatory processes at the scale of a skillset, or group of skills that all occur at the same phase of self-regulation. If we wanted to identify planning (but did not care what specifically individual students were doing), we could use a macro-level trace. For example, a macro-level trace might be a timestamp of a student interacting with a planner. Such a trace could only really provide an observer with the inference that a student is indeed planning. It could

not provide context on anything more specific, like how they decomposed a task or allocated time. These more detailed inferences require micro-level traces which focus on identifying specific skills within a particular SRL skillset. For example, analyzing how many different tasks they added to their planner could indicate how fine-grained a student was Decomposing a task. Alternately, seeing blocks of time allocated within the planner to certain tasks could indicate Scheduling. These traces can indicate more specific information — telling observers about particular components of the planning process. If an observer had access to traces about a student's proficiency to perform Task Analysis, Decomposing, and Scheduling, then they could use those inferences together to understand a student's overall proficiency with planning. However, in isolation, a single micro-level trace is too detailed to provide much information about a student's overall proficiency with a particular group of skills.

Besides scale, there is also the factor of how data traces are implemented. In Schraw et al.'s taxonomy [128] of measurement approaches in education, the authors separate measures that require a student's active attention (obtrusive measures) from those that do not (unobtrusive measures). An example of an unobtrusive trace can be found in Davis et al.'s work [22] where the authors collect a timestamp of when students complete a quiz with respect to the due date of that quiz. As it does not require a student to devote specific attention to engaging with the measurement, such data is classified as unobtrusive.

Within this work, we seek to identify traces that provide instructors with raw material to observe skills without bias. This means that traces that provide more detailed inferences are preferable as, hypothetically they could also indicate macro-level structures when used together. Thus, we focus on micro-level traces as they can provide the most detail and be used together to eventually make more macro-level inferences. We also focus on unobtrusive data traces as they capture authentic learning and cannot be biased towards a student's beliefs, like student self-reports.

A Note on Assessing Overall Quality

While a data trace that manages to be valid, reliable, and equitable at the same time would be ideal, that is unlikely in practice. Equally important, a given data source might not give sufficient information to properly understand a student's behavior with respect to a given skill. However, if we combine traces from multiple data sources, then we can hope to gain more insight about a given student's behavior. For example, an AAT may be able to tell us what reference tests a student's code submission failed, but we also need IDE-level tracking in order to see how they respond to that message. Help-seeking queues or discussion boards can capture a student seeking help, but do not tell us how long they worked independently before reaching out. Using both together allows for an inference that neither can make in isolation.

Using multiple traces also means that individual limitations on traces do not limit overall analysis. If each trace can provide a partial view, we need a variety of traces used together to paint a full picture. For example, as we just discussed, optional tools like ungraded practice platforms are not equitable. In isolation, if all we have are interactions with optional tools, our data is biased towards the stronger self-regulators who interacted with them. One could, perhaps, assume that all students who did *not* interact with the platform were struggling to regulate as effectively. However, when optional tools are assessed as one among a series of diverse traces, we could more effectively gather data about the class as a whole. We might be able to see that some percentage of those who do not engage with practice platforms spend that extra time starting assignments earlier, or reviewing feedback, or asking for help. Thus, a trace with strong reliability but weak equitability may still be a useful part of a greater whole.

5.2.2 Taking a Theory-First Approach

While there are no shortage of LA studies that focus on SRL [1, 22, 160], many reviews of the literature have been calling for a more theory-driven approach.

Bodily and Verbert noted in their 2017 review of LA literature that 14 out of 93 reviewed articles (15%) reported why they were collecting the data they chose to study [9]. Furthermore, Bodily and Verbert identified only 3 of those studies (3% of their full corpus) that "conducted a meaningful information selection process" [9]. Reporting no approach or justification to selecting data traces is an untenable practice. It prevents future researchers from replicating any approaches, limiting the capabilities of future work. Additionally, it makes assessing the rigor or quality of these studies impossible, which makes identifying good ideas from more questionable ones impossible as well.

Most troubling, it appears that this lack of reporting has lead to a concerning trend where studies do not ground their data traces or findings to existing educational theory. A lack of theory in the works reviewed by Matcha et al. was one of the key findings in their 2019 literature review [93] and Galaige et al.'s 2022 systematic review took, if anything, an even darker view on the state of theory within LA design [44]. Galaige et al. also reported the results of a survey to the LA community to identify current problems in the field, and a focus on technology over theory was a common thread among all of the problems identified. As the authors of that review summarize: "Contributing greatly to the unrealized potential of SFLA [Student-Facing Learning Analytics] are technocentric design methods that focus on the availability of data with little attention to learning science theory" [44].

Even if theory were to play a bigger role, focusing any approach that prioritizes availability over ideal traces is going to be limited. As Gray and Bergner put it:

> Reasonable validity and reliability in one context is unlikely to generalise to other contexts because working backwards from collected data to a measurement model is context specific...data collection should be preceded by identifying the learning constructs of interest and defining the measurement model.For educational technology, this means deliberately designing the collection instrument (and so the consequential trace data it collects) around constructs of the learning process [48].

For example, Gasevic et al. [46] note a situation where two biology courses make use of embedded assessments within the LMS, Moodle. Even though traces were relatively similar, their power to predict student outcomes was different. This was because in one class, the assessments were summative, meaning they were used as a way for educators to assess student progress against an expected benchmark. In this class, assessments could not be retaken. In the other class, the assessments were formative, meaning students were meant to use them to assess their own progress and were able to retake quizzes as often as they needed to. Gasevic et al. posit these differences in how the assessments were used could explain why interactions with quizzes in these two classes were not able to predict outcomes in the same way. Even if these two classes used the same quizzes with identical phrasing on the same platform, they were used in such different contexts that the traces cannot be generalized. Matcha et al. end up making the same recommendation, calling for traces to be derived from existing theory rather than availability [93]. What these authors are advocating for here is what I will call a 'theory-first' approach: where ideal traces are derived from an existing understanding of theory rather than availability of data.

Christea et al. take such a theory-first approach. The authors started with the four phases of self-regulation from the Winne and Hadwin model: task definition, goal setting and planning, enactment, and adaptation. They then identified interactions with their LMS that could indicate these four phases. While they do not show the full results of this step, they offer this example: "...task definition is defined as the perception of the features of the task students must carry out...Adapting this definition to our context, we can think of the student

trying to better understand the course content but also the assignments themselves" [20]. Once they have rough interactions with their LMS mapped to self-regulatory phases, they then moved onto data traces. When identifying data traces, they reviewed previous literature to see what traces had been used in other Learning Analytics studies. They also came up with a set of original data traces from their LMS that had not been previously studied, but seemed promising.

Beyond addressing the problems outlined already, I argue that a data-first approach limits what aspects of self-regulation we can study. Indeed, a major recommendation from my literature review from Section 4 was that future work would benefit from an increase in diversity, not just in what skills we study but how we approach studying them. In a data-first approach, researchers are focused within what they already have access to. It is my hope that the data traces discussed in this dissertation can act as a road map for future growth at any institution.

5.2.3 Approach to Deriving Data Traces

We started with the finalized list of SRL skills outlined in Section 5.1. While some initial behaviors had been identified for some skills, we seek to take a more formalized approach to deriving observable indicators for each skill. As previously stated, no agreed-upon framework yet exists for deriving data traces from skills, Therefore, our approach loosely follows the experimental approach laid out by Christea et al. [20]. This is because two of Christea et al.'s main priorities in identifying metrics were validity and portability between classes (two of the three criteria we value) and also seek to take a theory-first approach, using skills to derive traces. Using our list of skills, (defined in Section 5.1) we also leverage the COPES framework from the Winne and Hadwin model to help transition from abstract skills to

observable identifiers.

However, we differ in a few major ways. First, we derive traces from all of the different digital education tools that a CS class may use, rather than just the data gathered by an LMS. Second, while some behavioral indicators were identified in the Delphi Process and systematic review, many do *not* come from previous learning analytics studies. Matcha et al. [93], among others, note that while pulling traces from other papers is a common approach, it only works if those other papers derived traces in a way that has a solid basis in theory. Upon reviewing the papers Christea et al.use, few ground their traces back to SRL theory at all. Therefore, we focus more on identifying the best possible indicators for each of our identified SRL skills. Second, while Christea et al.focus on the Products (as defined in the COPES model in Section 2.1.2) at the *end* of each of Winne and Hadwin's four phases, we focus on identifying any and all Products and Operations that a student may use when demonstrating a specific SRL skill. This makes more sense given our focus on micro-level data traces. Overall, this means that the traces derived in this work are more broad in that they focus on the Operations and Products that a student might be using at some point while self-regulating, regardless of the technology.

Beyond those two differences, our process was the same. We start with the set of theory-grounded SRL skills outlined in Section 5.1 as most important to prioritize for success within CS. Table 5.2 and Table 5.3 detail the results of using the COPES model to identify indicators for each of our skills. Finally we mapped indicators to specific digital interactions, and arrived at a set of metrics that can afford a strong inference.

From there, we then assessed the technologies common to the digital ecosystem of a CS classroom and assessed the benefits and limitations of all of those data sources. The results of that evaluation is detailed in Section 5.3.

5.3 An Assessment of Data Sources

In this section, we consider tools that are commonly seen within the software ecosystem of CS courses through the lens of associated data traces and how they might be used to identify levels of SRL behaviors. We organize these tools into the following categories:

1. Practice Exercises
2. E-Textbooks
3. IDEs
4. Automated Assessment Tools
5. Discussion Boards
6. Office Hour Attendance
7. Specialized SRL Support Tools
8. Learning Management Systems

We also note limitations and biases that researchers need to be aware of when using these traces.

As explained in Section 5.2.1, each source will be evaluated in terms of the validity, reliability, and equitability of the traces that source can produce.

5.3.1 Practice Exercises

A wide variety of practice exercises are becoming prevalent in CS courses. These can range from small programming exercises [33, 104] to proficiency exercises that make students show the steps of an algorithm [71, 131] to basic batteries of multiple choice, fill-in-the-blank, and true/false questions.

Within CS classes, one common form of practice exercise is to ask students to write a small piece of code. That code is processed and students immediately receive automated feedback

regarding how correct their solution was. E-texbooks can integrate such exercises into their text content and plenty of sites exist that are purely for practice as well. Some coding practice platforms like CloudCoder [60] track keystroke-level data to see what students are doing as they write code. This provides a much more richly detailed source of information as it provides a rich view into a student's full process of solving a problem, including any misconceptions or misunderstandings.

Even in coding-focused classes, practice problems can extend beyond just typing in code. Many CS classrooms make use of Parsons problems, graphical practice problems where students must re-arrange mixed up blocks to form a correct answer. Often, these blocks are lines of code, and the correct answer is a working program [38], but they can take more abstracted forms and represent concepts like loops or conditionals that are separate from syntax [38]. As an example of what is possible, RuneStone [37] logs of when a student starts the problem, moves any line, and correctly completes the problem are tracked.

Practice exercises can also take purely conceptual forms too. OpenDSA [131], an e-textbook platform with practice exercise integration, makes use of interactive visualization questions. Students can interact with visual representations of algorithms or data structures to act out some operation. A student may need to click on the right nodes in a graph to demonstrate how a breadth-first search algorithm would act. Interactive slideshows that require a student to answer a conceptual question before being able to proceed to the next slide are also possible. With these problems, detailed logs of the state of the visualization are captured every time the user clicks on anything.

Evaluating Practice Exercises

There is an odd trade-off that one must consider when pulling data from practice exercises. Optional practice exercises offer a simple, yet highly valid inference because students only ever engage with optional tools when they decide that those tools are valuable for success. A trace that is nothing more than a timestamp of engagement still demonstrates that the student went through a process of evaluating the practice problems and determining them to be valuable to learning. That is advantageous as the trace is simple to collect and carries a straightforward inference to self regulated learning. What's more, such an inference is relatively reliable when shifted between different class contexts as well, whether it be a senior-level post-secondary class or an early high-school class, the timestamp of interaction still strongly indicates a student found the optional tool useful.

However, any optional tool gains that validity and reliability at the cost of equitability in the population sampled. Null data from such systems encompasses both students who effectively self regulated by deciding that the tool was not helpful for them, *and* students who performed no such assessment. One of those populations successfully self regulated by assessing the tool as unhelpful, the other might include students who are still learning to self regulate effectively and do not yet have the tools to perform that assessment. Thus, any practice exercise that is able to be skipped focuses only a subset of students who are strong, self-regulated learners.

Requiring engagement with practice exercises has the opposite problem. Once engagement with practice problems is no longer because a student chose to practice, we gain a more equitable data set yet sacrifice that inference. This trade-off is discussed more at length in Section 5.3.7, where all of the tools are optional.

Fortunately, unlike with the self-regulation tools in Section 5.3.7, required practice problems

still offer effective data traces. Keystroke-level data (as collected by CloudCoder [60]) provide a complete view of how a student coded their solution from their first attempt until their final submission. Such data could indicate when a student got stuck, what misconceptions stalled their progress, and what they prioritized within the problem solving process.

A lot can also be learned from practice exercises about whether a student is doing them only for credit, or as an aid to learning. "Gaming the system" is a well-known problem with any educational software where a grade is required [7]. Identifying gaming is a clear indication of an SRL anti-pattern.

Overall, some practice problems likely collect enough context about a given work session that they still lend themselves to highly valid data traces. Reliability, however, remains a limitation as that detailed data also is likely highly contextual to the grading requirements of a class. For instance, a class where practice problems are graded are going to have broadly different engagement patterns compared to classes where the same problem sets are ungraded. Classes where test problems are clearly rephrased practice questions might see different engagement patterns from classes where the same questions are posted as optional challenges for advanced students. Thus, practice exercise engagement, even accounting for the same questions on the same platform in a class covering the same content, could vary wildly depending on other course-specific variables.

5.3.2 E-Textbooks

Within most CS courses, where learning takes place both in a classroom and digitally, content is stored on a digital platform for students to access. Logs of when a student accesses pieces of content are valuable data traces, and these tools typically capture such data.

At their most basic, e-textbooks are nothing more than a digitized version of a paper book;

allowing students to access readings from their laptop, rather than a library. Tracking data when a student opened a page of text, tracking them scrolling through the text, and knowing how long that page remained open before navigating away can help outside observers deduce that the student was indeed reading. Contextualized with what topic the student was reading about could also help inform inferences about what they prioritized in a particular learning event.

Broadly speaking, however, digital textbooks tend to take better advantage of their medium and often do things that a more traditional book could not. For example, both OpenDSA [131] and RuneStone [37] augment prose with embedded media. Things like slideshow demos where students can get a step-by-step visualization of how an algorithm works or video demos allow an e-textbook to support learning through a number of different channels. Practice problems (discussed in Section 5.3.1) are also commonly embedded into e-textbooks.

There are several platforms that integrate student data to create personalized experiences for students. Another textbook platform, ELM-ART [154] can go even further, adapting the order or pace of content depending on the student's needs.

OpenDSA and RuneStone also log information about a student's interaction that paint a detailed picture of how that student worked in a learning session. Logs of when a page was accessed, when practice problems were attempted (and what the results of that attempt were), when students engaged with visualizations, and when students watched video are captured in both of these platforms.

Evaluating E-Textbooks

Inferences using the data available from e-textbook systems have a relatively high level of validity to them. Accessing a page (at least for more than a trivial amount of time)

will usually imply the student is actively working to learn that material. In classes where associated practice problems are ungraded (or when the student repeats an already graded exercise, such as before an exam), engagement within these systems can indicate that a student is monitoring their own understanding of the material with relative validity as well.

Data from e-textbook systems also tend to be equitable. Assuming that an e-textbook is the only venue where students can engage with this material, these systems do a good job of tracking behavior from all students, not just those who are particularly strong at SRL. Even if the content from an e-textbook was available elsewhere, the data captured would not be biased towards strong self regulators, but rather toward those that prefer reading online.

However, the data collected may not reliably collect good inference data when moved to other class contexts. Even when two classes use the same textbook, there are other contextual variables in a class that can influence how a student engages with that material. A class with weekly reading quizzes or where engagement with exercises is graded is going to see wildly different forms of interaction than a class where the textbook is a fully optional supplement to lectures. Thus, researchers will need to account for class context when trying to understand these data traces.

Another limitation of using traces from e-textbooks is that they lack nuance. We can see the 'when' and the 'what' but have to infer a lot of the 'why'. Without sufficient context that we typically do not have, we end up needing to make a lot more guesses about a student's intention, which ultimately results in less complex inferences. This ends up harming both validity and the scope of what we can discover about our students' learning behavior. For example, erroneous clicking becomes hard to separate from intentional selection of a resource.

5.3.3 IDEs

Integrated Development Environments (IDE) allow students to edit, compile, and run a program, and get feedback on results and various code quality metrics. An IDE is a nearly ubiquitous tool within any class that requires programming. What's more, they offer an opportunity to gather highly detailed information about a student's learning experience. While these systems do not often log data traces on their own, other integrated software tools can capture extremely detailed metrics. For example, DevEventTracker [67] (a plugin for the common IDE, Eclipse) logs keystroke-level data of a student's actions while coding, like the practice platforms discussed in Section 5.3.1. Additionally, this IDE-level tool tracks timestamps of when a student is working, what errors or problems arose when they compiled their code, and information regarding how effectively the code has been tested. The main advantage that IDEs have over practice platforms is that IDEs are significantly more powerful and richly featured, making them a better choice for students working on more complex, longer-term assignments. Conversely, practice platforms' focus tends to be relatively small in the scope of the problems they ask. Thus, it is not only possible to capture how a student is learning within small practice problems, it is possible to track their progress end-to-end on large assignments.

Hackystat [64] takes a slightly different approach, focusing on time rather than snapshots of code. This system tracks active time and keeps track of the most active file. File size and complexity, unit test results, and test coverage are also noted. Thus, while far less detailed than knowing every keystroke a student entered, Hackystat allows researchers to make a strong inference about when a student was working and what files they worked on.

Evaluating IDEs

In both the keystroke-level tracking of DevEventTracker and the more meta-data focused approach of Hackystat, data traces are likely highly contextual to the class and the assignment, meaning that reliability could be an issue.

Equitability, on the other hand, is strong because students are often required (or heavily encouraged) to use a particular IDE within a CS class (especially when they are learning to program). This means that IDE tracking does not bias the population towards only strong self-regulators.

The validity of data traces depends somewhat on the type of data collected. The detailed data of DevEventTracker gives a great deal of context into what a student was coding at any given moment of their work session—leading to likely highly valid inferences. Hackystat could potentially struggle to capture as much nuance, meaning the inferences made from data need to be more simple to ensure validity.

5.3.4 Automated Assessment Tools (AATs)

By their very nature, CS classes frequently require students to complete programming assignments. Programming is an inherently iterative activity. The act of programming is involves making an educated guess about how a unit of logic should work, and then checking by validating that guess. CS Educators often encourage students practice incremental development [68] as a formal process.

It is therefore no surprise that the way programming assignments are often set up to support iterative evaluation. Unlike other forms of assignment, where a finished product is submitted once and graded, CS students often have the ability to submit their code to an automated

suite of tests and receive a grade and feedback. Notably, only their final submission's grade is saved, meaning students can check their work more formally as they code.

AATs are tools that automate some form of programming assignment evaluation. They typically run a suite of automated software tests on a student's code to determine if the submitted work matches requirements. Because the systems do not require attention from course staff, students are able to check their work on their own time. In most CS courses, that paradigm encourages students to submit in-process work early and often in order to check on their correctness.

While some AATs like TestMyCode [105] do track keystroke-level data, more commonly these tools only have access to a 'snapshot' of code from when it is submitted. These snapshots are often stored using souce control tools, like Git, meaning tracking changes over time is relatively easy. To name a few, Web-CAT [34], ProgEdu [16], and Marmoset [140] take this approach and commit all code to Git repositories. PruTutor [21] and Edgar [94] even transform those snapshots into visualizations to show back to students, much like an LA dashboard. As these systems are built to grade and offer feedback on a student's submission, data regarding how the submission scored is typically stored with each code commit. These snapshots captured by AATs sit somewhere between DevEventTracker's detailed keystroke-level data and Hackystat's metadata tracking. AAT-level code snapshots are intermittent landmarks on a student's journey towards a fully correct answer. They still can tell the story of how a student progressed from starting an assignment to finishing it and can still afford similar inferences. However, some of the nuance of how they reached a particular decision is lost. Students sometimes will not make a first submission until they think they are fully finished. This means that these snapshots are less effective at observing how a student starts a project and more effective at observing how a student finishes.

AATs also offer feedback to students that they cannot receive from other development envi-

ronments (like an IDE). When a student submits code to an AAT, a suite of reference tests evaluate that submission and offer feedback. Typically, that feedback relates to how correct that work is. However, a few AATs (such as Karnalimet al.'s CCS [66]) can also evaluate the quality of a submission in other ways such as how effectively a student has tested their own work, how well-documented the work is, or how stylistically correct it is [95]. This can be valuable context to take into account when assessing work to see if a student effectively regulated their time. As described in Chapter 4, assessing time-management by looking at how close to a due date work was completed can be problematic, as it can be difficult to differentiate a student who is busy but skilled at allocating time to complete work from a student who allocated time poorly and is rushing to finish. However, those two students would likely have noticeably different code quality metrics in their submissions. Thus, such traces, when available, could be especially valuable for better understanding submission patterns.

Evaluating AATs

Reliability between class contexts remains an issue for AAT-based measures. Adopting standardized formats for data is a logistical hurdle that all of the data sources discussed in this section need to surmount (see Section 6.2 for more).

Even once the data is stored in a standardized way, class context can cause the way students engage with an AAT to vary widely. Therefore, researchers can not expect that data collected in two different classes to lead to the same SRL inferences in the same way. For example, Arakawa et al. [3] used the number of submissions made to Git (which had built in testing and was used like an AAT within the study) to identify at-risk students. Such a metric might be highly unreliable for institutions using Web-CAT which can cap the number of submissions per hour to limit spamming behaviors [63]. Rate limiting submissions changes the total number of submissions possible in a way that Arakawa et al.'s study likely did

not allow for, meaning the metric could not be used in another class context without some adjustment. For example, the threshold for the number of submissions to identify at-risk students might be lower in a class that allowed fewer submissions to an AAT overall.

Another grading issue that can have a huge impact on behavior is whether test suite quality is graded, and how. When test suites are not required, students are less likely to do organized test suite development. Grading test suites by code coverage might reveal gaming behavior as they seek to maximize test suite points without actually writing a good test suite [133]. Mutation testing [23], as an example of a strong test suite metric, ensures that such maximization efforts do not work. However, they might not avoid gaming the timing of when the test suite is developed.

5.3.5 Discussion boards

Often, classes will make use of a private, text-based forum for course-specific conversations.

Discussion boards are frequently used as a tool for students seeking social help from their peers or course staff. Help-seeking skills are an important self-regulation skill [30], but they are also difficult to observe [31] directly. Especially among novices, a sense of belonging is a major influencer in how students seek help [28] and direct observation is often avoided since it could make students more hesitant to ask 'dumb' or personal questions [125]. Thus, having an unobtrusive view into how students seek social help may be one of the best ways to get a better view of this important skill. A common feature within CS classes is a help-focused discussion board where students can ask questions and receive responses from instructors or their peers. Combined with the content of the post, this could offer us a view into when a student asked a question and how they formulated it. More feature-rich discussion board platforms have gone further, logging events when a student views or upvotes a post as well.

This could allow us to know that a student is seeking help even when they do not directly ask a question.

Some researchers are already beginning to make use of data traces from discussion boards to examine help-seeking as well. Thinnyun et al. [150] used metrics from Piazza [127], including the number of questions a student asked, the number of answers that same student gave to peers, and the number of days between a first and last post, in their study on gender participation in social help-seeking.

Discussion boards can also be used as a venue for other types of asynchronous communication. They can be a formal venue for discussion-based assignments where students need to reflect on a topic related to the class [147]. Like any other form of digital assignment submission, a discussion post is timestamped. This means that they could potentially offer insights about when, in relation to a due date, a student finishes their work.

Evaluating Discussion Boards How students use discussion boards can vary between class contexts, but a student seeking help will look fairly similar across those contexts. So long as two classes make use of such a tool, a discussion post with relevant text will be relatively indicative of help-seeking regardless of class context or policy.

When we see a discussion post asking a help-seeking question, we assume the student was indeed seeking help in some way. While this is a valid inference, it does rely on some analysis of the text of the question to make sure the post is actually asking a question. For example, if researchers counted all discussion posts as help-seeking events, they would potentially end up counting all textual answers to every question as help-seeking events erroneously. While developments in machine learning and natural language processing have made this easier [65], to some extent the class culture will influence the ways in which students ask questions, or indeed who asks questions to begin with.

Equitability is also somewhat context dependant. Discussion boards that require associating a question with a specific name (rather than allowing students to anonymously post) will likely be biased towards students who feel comfortable enough asking a question in front of their peers. As novice programmers sometimes view help-seeking as a sign of weakness [28] and can be hesitant to appear foolish in front of peers [125], this can strongly influence the choice of which students choose to use this tool.

5.3.6 Office Hour Attendance

Visual queuing systems like MyDigitalHand [138] or HelpMe [153] can keep students organized and can be useful for students seeking assistance in office hours. Zoom [167] has a similar feature, noting when a person enters a waiting room, main room, or breakout room. These similar features mean that video calling systems could provide analogous data to a more formalized visual queue. By tracking when a student enters a queue, when they leave the queue and receive help, and when they leave the platform entirely (after receiving help), these tools unobtrusively give researchers information on when exactly a student receives help and how long they are willing to wait for course staff to receive that help. In short, the tools exist to view help-seeking in a variety of different modalities.

Some systems can also require students to complete a custom survey before receiving help from course staff. This can offer some insight into what a student is seeking help on and what kinds of questions they might ask, though these are likely more student self-reports than data traces.

Evaluating Office Hour Attendance Noting when a student enters office hours is a trace with high validity. Students rarely enter office hours by accident, and almost always come because they are seeking help. This makes the inference from "office hour enqueue

event" to "student is seeking help" very strong. This can help us know not just how frequently a student sought help over an assignment's timeline, but how close they sought that help to an assignment's due date.

Enqueue events for office hours likely also translates between different class contexts fairly well. Regardless of the question asked, the fact that a student came to office hours, and the time they did so likely mean the same thing in different contexts.

Variations in course content and structure limits equitability in the same wyas as Discussion Posts. Office hour attendance relies on a lot of factors beyond simply the need for help. Depending on the assignment's difficulty, the class's overall perception of seeking help, and a student's rapport with course staff, students could have a wide variety of reasons for avoiding office hours. Thus, as with discussion posts, it is important to keep in mind that these systems gather data from those most comfortable with speaking up.

5.3.7 Specialized SRL Support

This category encompasses a suite of tools that are there specifically intended to help scaffold SRL. These are systems that are added onto other digital platforms that help students practice specific self-regulation skills. Van Der Graaf et al. [152] note things like built-in timers, support for highlighting text or taking notes within a digital learning environment, a search function for relevant content, and a digital planner as examples of these tools. They are not typically a required component of engagement within a class, but they are on hand in case students find them valuable.

For the purposes of our work, these are optional tools built into digital education software that could be used to help students self-regulate. For example, if a timer was a built-in feature of a e-textbook system, a student could designate a specific study time and use the

remaining time to monitor their progress through a reading or a piece of work. In addition to helping students practice self-regulation skills, these tools also can gather valuable traces. When a student chooses to engage with an optional tool like a timer, researchers can make a relatively strong inference that a student is indeed self-regulating their time.

The field of Learning Analytics has had no shortage of such tools. Azvedo et al. [5] proposed MetaTutor in 2009. This software acts as both a learning environment and a tool for scaffolding self regulation by offering a structured system for students to evaluate their confidence in their understanding and, in more recent iterations, a built-in planner [148]. Within CS, Edwards et al. [35] also created a built-in planner by creating a digital version of Spolsky's painless schedule sheets tool [142], as part of an intervention on procrastination. Winne et al. [160] proposed nStudy in 2019 which offers many of these types of tools. This extension to the web browser Google Chrome allows students to bookmark a page as important, highlight and take notes, and search among recorded data for specific terms. The tool also contains several different views with which users can engage with these tools and perform searches. nStudy captures traces whenever a student engages with the tool in any way, noting both when the event occurred and some content pertinent to the type of event. While these tools predominantly track the ways students monitor and control learning in process, they are not limited to just that area of self-regulation. Hsiao et al.([61]) created a unique view of assessments that offered students buttons to press in order to bookmark specific questions or note when they now understand a missed question.

Evaluating Specialized Support Tools Inferences from specialized tools are perhaps some of the most valid we have discussed in this paper. When a student engages with a planner, we can indeed make a strong inference that that student is attempting to plan out their work. Systems like these are almost always optional, meaning that one could anticipate

that two different classes could expect relatively similar engagement patterns out of their use. This makes these scaffolding tools highly reliable as well.

However, as is the case with any optional tool, these systems also capture student behavior unevenly. Such software only can capture data on the students who *choose* to engage with it. As Van Der Graaf et al.put it: "...there are several factors that can affect tool use, such as whether learners are aware of the tool, whether they think the tool can be useful to the given task, and whether they have skills to use it" [152]. That choice is part of what allows researchers to make such a strong inference, since it implies a student evaluated that using the tool would be useful.

Thus, these tools offer a trade-off: better inferences than interactions with required software, but worse insight into the students who would most benefit from the tool's use.

5.3.8 Learning Management Systems (LMSs)

LMSs like Canvas [80] or Moodle [97] could have been brought up in nearly every previous section. LMSs have a variety of built-in functionalities that accomplish many of the same roles as the other tools discussed. Additionally, many modern LMSs directly integrate with external software tools through the Learning Tool Interoperability (LTI) Standard [19]. A class using an LMS could ultimately be using any variety of external tools as more of a central point of connection for the different types of tools discussed in this section. In this way, an LMS is partially an e-textbook, partially a practice platform, partially discussion board, and so on.

Yet an LMS is also none of those things as well. LMSs are defined by being a jack-of-all-trades product rather than by a specific function. While an LMS does contain support for presenting content and practice exercises online (much like an e-textbook), a LMS does not

cease to be valuable if a given class does not wish to use this functionality. The same thing is true for discussion boards or automated assessment tools or any other system discussed here. It is therefore better for our purposes to consider an LMS as a centralized hub of data, rather than a distinct tool in its own right.

This makes analyzing validity, reliability, and equability for LMSs much harder. In some respects, they are as precisely valid, reliable, and equitable as any tool they make use of.

5.3.9 Summarizing Findings

Table 5.4 summarizes our assessments of how each of these tools rates on Validity, Reliability, and Equity.

	Validity	**Reliability**	**Equity**
Practice Exercises	Depends on course context	Depends on course context	Depends on course context
E-Textbooks	High	Medium	High (assuming it is the only place information is stored)
IDEs	Depends on how data is analyzed	Low	High
AATs	High	Low	High
Discussion Boards	High	High - Events of a discussion post that contain a question Low - Qualitative aspects of a given question	Medium
Office Hour Attendance	High	High	Medium
Specialized SRL Support Tools	High	Depends on course context	Low

Table 5.4: Summary of Validity, Reliability, and Equity Assessment

5.3.10 The Value of a Multi-Source Approach

There are two major advantages to be had from integrating data traces from a wide variety of different sources.

First, as we discussed previously, by capturing behavior from engagement with a variety of different platforms, we better understand the context of what a student is doing at any given moment. An AAT may be able to capture when a student is struggling with a bug in their code, but it cannot capture how a student responds to that struggle effectively. Using multiple sources together allows for an inference that none can make in isolation. Furthermore, in isolation, events of when a student opened a page are difficult to tie to help-seeking. However, seeing a student repeatedly re-submit code with the same AAT error feedback and then go look at a page of content tells us a great deal more. This also has the advantage of helping with data cleaning and pre-processing. As described in Section 5.3.2, researchers currently have to remove events where a student erroneously clicked, or when they have moved on to other things while leaving a page open. These idle periods are blank spaces in our understanding of what a student is doing. We can fill in some of those gaps if we have data from other sources. For example, we would be able to identify when a student is working *while* they have the instructions open, meaning that even if the student is relatively inactive in click events on the page, they might still be actively using a page. This additional context could allow us to determine a more reliable (and potentially dynamic) heuristic for determining when a student truly goes inactive. Plus, the added context of what a student was doing before or after a particular interaction informs the inferences we make. Erroneous interactions are easier to rule out, leading to increased validity in our traces.

Second, as we touched upon in Section 5.2.1, including many traces from sources that are each limited in different ways means we can mitigate those individual limitations. For

example, specialized SRL support tools offer highly valid inferences, but do not holistically capture behavior from a whole class. In isolation, this could only tell us that Student X used the planning tool and planned to work at 6:40 the following evening. For example, it would be a fairly safe inference that Student X is indeed effectively scheduling time to work—an important SRL skill. If Student Y chose not to use the system, we have no means of measuring their SRL ability. However, with IDE tracking like DevEventTracker or AAT submissions, we can track not only how Student X chose to spend that work session, but how Student Y ended up starting and finishing their work on the same day. Even though Student Y opted to not use the planner, we can see when and how they chose to spend their time working. Thus, strong data traces, like using a planner, adds strength to our overall observations, but does not limit our observed population to only those who used that optional tool. We gain the added benefits of tracking all students and making relatively strong inferences without the biases of any individual source.

5.4 Data Traces for High-Priority SRL Skills

The second research question of this dissertation was: How do we measure self-regulation skills effectively?

We answer this question most succinctly in Table 5.5, Table 5.6, and Table 5.7. These tables present our list of Skills, Operations, Products, data traces, and potential data sources for those traces. These tables should, ideally, act as a road map for future researchers wishing to empirically study self-regulation within CS classes.

There were several cases where we could not see how our Operations and Products could be measured by unobtrusive traces alone. These are noted in gray as they could still be valuable to future researchers looking to use both traces and self-reports. For instance, data alone

can not measure a student's reasoning behind taking an action. Thus, unobtrusive traces do not provide the context to differentiate a student changing locations because they assessed that one was better for working as opposed to because the library was closing for the night. In order to know why they made a choice, we would need to ask directly, necessitating an obtrusive self-report. Such measures are outside of the scope of this work.

While we have tried to be thorough, we also recognize this list may be preliminary and other research teams may see other traces that would fit in this table.

Skill	Operations	Products	Data Traces	Data Sources
Task Analysis	Inferring requirements that were not clearly stated in the instructions	Work contained inferred requirements	Qualitative attributes of work produced	Keystroke-level capture of code (from IDEs or practice exercises) Submission-level capture of code (from AATs)
		Record of student seeking clarification	Timestamp of when question was asked	Discussion board
			Text of question	
	Forming a correct conceptual model of the task	First draft of work demonstrated a correct model	Qualitative attributes of work produced	Keystroke-level capture of code (from IDEs or practice exercises) Submission-level capture of code (from AATs)
		Solving test cases before beginning	Timestamp of first interaction with test case	Platform that was storing test cases (possibly IDEs, practice platforms, or AATs)
			Timestamp of first correct solution submitted for test case	
		Having a first submission that matches requirements	Attributes or reference test scores of first graded submission	
	Reinterpreting materials to make sense of the task	Took notes on instructions	Text of what passage a note might be referring to	Note-taker or other SRL support tool
			Text of notes	
			Timestamp of when notes were taken	
		Highlighted important components of instructions	Text of what passage was highlighted	Note-taker or other SRL support tool
			Timestamp of when the highlight took place	
	Reading assignment thoroughly	Spent active time with assignment open (before getting started with work)	Time delta between closing and opening the next assignment page	Derived from a page access event and the next subsequent page access event (e-textbook or other content display tool)
	Searching for relevant information	Accessed where that relevant information is stored	Timestamp of access to material	Page access of content (e-textbook or other platform)
			Duration of engagement with material	
	Identifying unclear instructions and seeks additional information from instructors	Written record of student seeking clarification	Presence of some clarifying question	Discussion board
			Timestamp (or presence) of engagement with another student's clarifying question	
			Text of clarifying question	
Scheduling	Allocating time in a deliberate manner	Wrote plan for time allocation	Timestamp of interaction	Planner or other SRL support tool
		Used a tool (like a planner or calendar)	Timestamp of interaction	
	Checking in on due dates before beginning	Accessed information	Timestamp of view	Page access of course logistics (commonly stored in LMSs)
Decomposing	Articulating a set of sub-tasks	Outline, to do list, or other sketch of work to be done	Timestamp of creation	Planner or other SRL support tool
			Time spent creating artifact	
			Qualitative attributes of artifact	
	Focusing in on a single sub-task	Different drafts of work focus on specific goals	Qualitative attributes of code from a single work session	Keystroke-level capture of code (from IDEs or practice exercises) Submission-level capture of code (from AATs)
	Student applied some form of prioritization to their list of sub-tasks (they picked some item to start first)	Student works on one feature of the assignment	Qualitative attributes of code from a single work session	Keystroke-level capture of code (from IDEs or practice exercises) Submission-level capture of code (from AATs)

Table 5.5: Operations, Products, Data Traces, and Sources for SRL skills related to Planning

Skill	Operations	Products	Data Traces	Data Sources
Monitoring Correctness	Validating work completed before continuing	After completing some unit of work, that unit is edited and evaluated before progressing	Timestamp of validation process starting	Keystroke-level capture of code (from IDEs or practice exercises) Submission-level capture of code (from AATs)
			Text (or other relevant record) of what was changed and updated in the validation process	
		Rubric (or other assessment tool) open while working	Timestamp of access to relevant information (contextualized with timestamps of work)	Page access of course logistics or Records of engagement with assessment (both commonly stored in LMSs)
	Assessing correctness on individual test/assessment questions	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
Monitoring Progress	Seeking extension	Communicating need for extension with course staff	Timestamp of extension request	Discussion board
			Qualitative attributes regarding how the of extension request was phrased	
	Comparing progress to their own expectations	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
	Comparing progress to due date	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
	Comparing their progress to the time they have allocated	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
Emotional Awareness	Demonstrating patience and an internal locus of control	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
	Demonstrating resilience towards failure	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
	Taking a deep breath	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
Knowing How To Seek Help	Asks specific questions regarding work	Written documentation of questions asked	Presence of some question	Discussion board
			Timestamp (or presence) of engagement with another student's question	
			Text of question	
	Utilizes a variety of resources to get answers	Records of access to that variety of resources	Timestamp of access to material	Derived from a page access event (content display tool, ex.e-textbook)
			Duration of engagement with material	
			Qualitative attributes of material	
	Coming up with a hypothesis of where a misunderstanding is	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
	Using pre-defined strategies to methodically help them find the point of confusion	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
Knowing When To Seek Help	Tries some number of strategies before seeking help	Records of strategies used before going to office hours	Timestamp of access to material	Page access event to content (e-textbook) or discussion board post or page access of course logistics
			Duration of engagement with material	
			Qualitative attributes of that engagement	
		Spent time working before seeking help	Time delta between getting stuck and seeking help	Getting stuck - identified from either keystroke-level capture of code (from IDEs or practice exercises) or submission-level capture of code (from AATs) Seeking help - identified from discussion board post event or office hour enqueue event
	Seeks social help when needed, but not constantly	Frequency of office hour attendance	Timestamp of when a student entered office hours (contextualized by qualitative attributes of work)	Office hour enqueue event contextualized with either keystroke-level capture of code (from IDEs or practice exercises) or submission-level capture of code (from AATs)
			Timestamp of when a student was seen by course staff	Office hour dequeue event
			Time delta of student/course staff interaction	The time between when a student is dequeued from the office hour queue and when they leave the system entirely.
Reducing Distractions	Starting a learning session in one location and moving to another location	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
	Assesses the qualities of a desirable location for work (possibly can articulate them)	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A

Table 5.6: Operations, Products, Data Traces, and Sources for SRL skills related to Monitoring and Control

Skill	Operations	Products	Data Traces	Data Sources
Reflection Using Internal Standards	Students able to articulate their standards	Removed as it requires a self-report and is therefore obtrusive	N/A	N/A
Reflection Using External Standards	Student revisits rubric	Student reviews rubric after working	Timestamp (or presence) of engagement with page where rubric is stored	Page access of course logistics (commonly stored in LMSs)
	Internalizes feedback from returned work	Makes changes on an assignment after viewing previous assignment feedback	Qualitative attributes of work produced (contextualized by a time delta of how longstudent viewed feedback for)	Keystroke-level capture of code (from IDEs or practice exercises) or submission-level capture of code (from AATs), contextualized with page access of course logistics (commonly stored in LMSs)
	Clarifies unclear feedback	Record of student seeking clarification	Presence of some clarifying question	Discussion board
			Timestamp (or presence) of engagement with another student's clarifying question	
			Text of clarifying question	
	Picking up or reviewing feedback	Engaging with an exam after it is graded	Timestamp of feedback access	Feedback page access event
			Time delta of feedback view	
		Engaging with an assignment after it is graded	Event of making changes to an assignment (contextualized with when the grades for that assignment were released and when the due date was)	Keystroke-level capture of code (from IDEs or practice exercises) Submission-level capture of code (from AATs)
Exploration	Investigating	Student asks tangential questions	Removed as it requires a self-report to differentiate between truly tangential questions and misunderstandings (and is therefore obtrusive)	N/A
		Tinkering	Removed as it requires a self-report to differentiate between changing code with the intent to investigate and misunderstandings (and is therefore obtrusive)	N/A
	Practicing	Engaging with optional practice problems	Event of engagement with practice problems	Practice problem host (devoted platform or e-textbook)
			Number of attempts made on practice problem	
			Timestamp of first correct submission of a practice problem	
	Practicing outside of the scope of the class	Engaged with already completed practice problems	Timestamp of engagement with problem (contextualized with first engagement)	Practice problem host (devoted platform or e-textbook)

Table 5.7: Operations, Products, Data Traces, and Sources for SRL Skills related to Reflection and Exploration

5.5 Examining Help-Seeking

Thus far, we have presented a menu of data traces, grounded in SRL theory, and discussed possible sources for those traces. Now, we demonstrate how a collection of data traces could be used together to get a better view of a student's self-regulatory behavior. In the literature review described in Section 4, we note that help-seeking remains a particularly difficult self-regulatory skill to examine, but also one that is highly important to success in CS (as demonstrated in our Delphi Process study described in Section 3). While some help-seeking behaviors remain hard to recognize, using traces from a whole digital ecosystem could be leveraged to improve our view of the help-seeking process for individual students.

Figure 5.1 shows a state diagram of the phases a student goes through when seeking help when they encounter a problem while programming. A student will be coding and eventually come to a point where they are struggling to make progress towards a completed solution and start seeking out resources to help them. Those resources could be entirely digital, like re-examining the instructions or going over a misremembered concept in a textbook. Doebling and Kazerouni [28] observed in 2021 that students tend to then progress to more social forms of help like asking peers or course staff to get help. Whatever sources they start with, they will eventually come across some potential solution to their problem and try it. That solution will either solve the problem (meaning they will progress towards a solution) or it will not (meaning they continue seeking help from various resources).

At each stage of this process, software already exists with which to capture such data traces. In some cases (like identifying when a student becomes stuck), there are multiple existing sources for such information. This makes capturing data traces related to help-seeking increasingly feasible. Even if researchers do not have access to every data source discussed in this paper, they could still likely capture some of this process using the tools already in

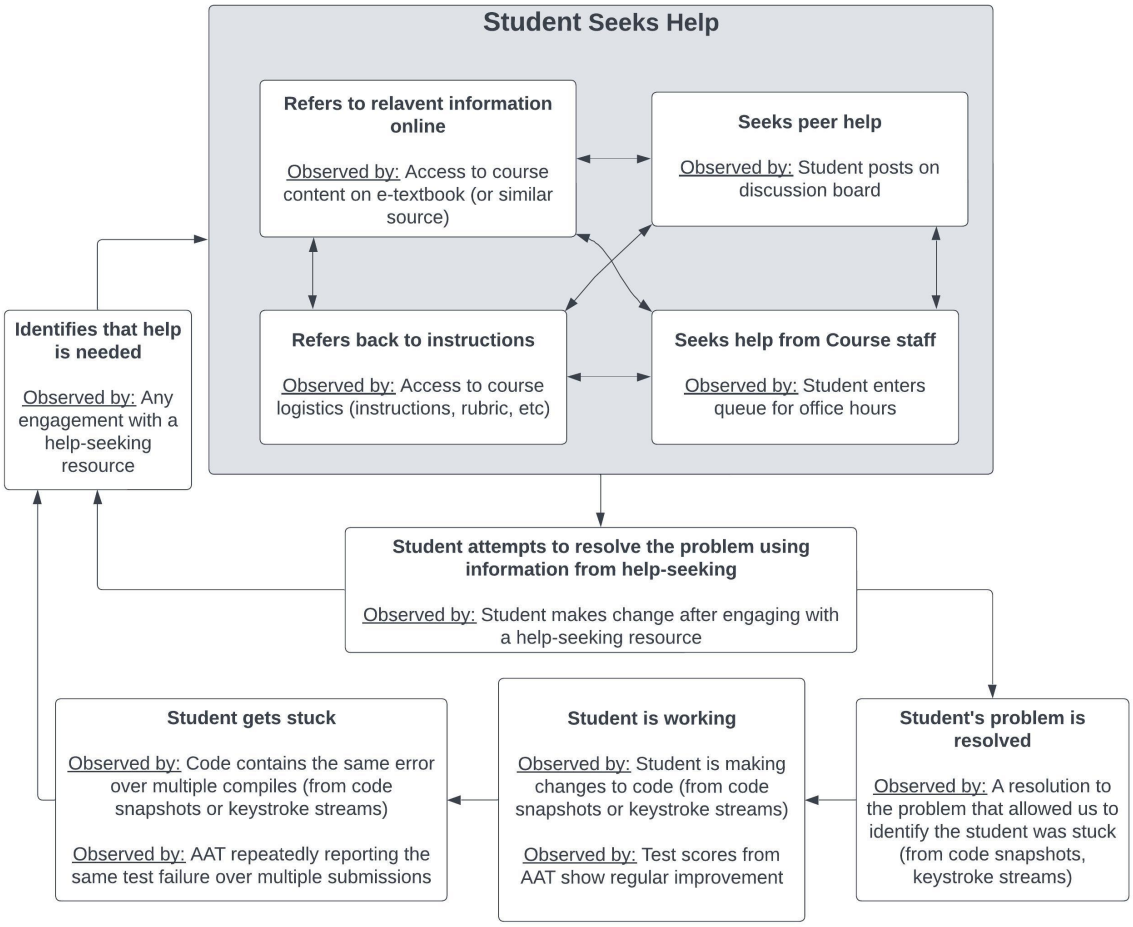

Figure 5.1: State Diagram of Help-Seeking Processes

place.

Chapter 6

Additional Challenges

In this section, we explore some challenges for future researchers seeking to extend this dissertation's work. First, validating that the data traces identified in this dissertation do indeed allow for good inferences back to the internal structures we want to be studying. Second, aggregating a set of software tools that all store data in distinct ways into one holistic database of events. To some degree, this second problem could be solved soon as improvements to data standardization are present, even if they are not fully adopted at time of writing.

6.1 Validation

The most critical challenge ahead for this work is to start the process of validating all of the data traces proposed here. Assuming all data is collected, normalized, and cleaned, future research then must contend with the problem of proving that the traces we have identified really do capture the Operations and Products we want them to.

This will be an especially hard problem to face as self regulation is so internal. Within current literature, a ground truth is often established by checking metrics against an inventory of self-regulatory skills [20], retention [1], or academic outcomes [4]. While these approaches are likely a good start towards validation, I ultimately believe these data traces will require validation from a number of perspectives. This is because all forms of measurement are, at

best, approximations and best guesses. There is no one truth we can use to prove that our traces are measuring the skills we want them to measure. Thus, many sources that work together to triangulate what self regulation looks like is going to be more accurate than any source could be in isolation.

For calibration purposes, having some sort of comparison against student self-reports will likely also be necessary (though I imagine it will require a fairly generous margin of error given the flaws with student self-reports outlined in this paper). Similarly, since self regulation is such a strong predictor of academic outcomes [17], I recommend using a variety of academic outcomes (homework assignment grades, exam grades, mid-semester and final grades, number of students who passed versus those who withdrew or failed, etc.) as well. Self regulation, self efficacy, and motivation are so closely tied together, assessing for these related constructs is also valuable.

6.2 Associating Records

In order to see student interactions across a variety of platforms, we need a way to identify that a given set of traces do indeed come from the same student.

However, educational technologies were not designed to share data, and it is a known issue that educational technologies do not integrate well [12]. One major consequence of this is that there currently is no uniform way to associate all records from a single student together. For example, some of the common digital tools used within Virginia Tech's CS courses are: Web-CAT (an AAT) [34], OpenDSA (an e-textbook) [131], Piazza (a discussion board) [127], and Canvas (an LMS) [80]. However, each of these sources uses a different system for assigning an ID. A student could be internal ID number 4 on Web-CAT, 301 in OpenDSA, S2234 in Canvas, and 5280 in Piazza. At time of writing, the only way to

determine if these four records go together is to use the student's name and email address, but these aren't consistent either. A student could be John Doe in Web-CAT, and John D. Doe in Canvas and OpenDSA, and Johnathan Doe in Piazza. Similarly a student may have used their personal email for one of these accounts, meaning two go to j.doe@vt.edu and one to DoeJohn@gmail.com.

This presents a problem with data at larger scales. As an example of this issue, in the summer of 2023 I created a script to create a unifying ID for records for these software systems (OpenDSA, Web-CAT, Piazza, and Canvas) that was tested using data from two semesters of course data from a single class (CS2114). Even when the script could create a unifying ID number for most students, there were several who needed to be assigned an ID by hand because of discrepancies in what names and emails they used with each platform. Thus, even with some automation, determining if two records are indeed the same person did require some level of human oversight. For two semesters of a single class, that involved processing less than 10 students. However, it is possible that at the scale of 2-3 classes, or 2-3 semesters, the required time to process entries that a script could not handle could amount to a significant number of human hours.

6.3 Privacy

This difficulty in unifying IDs also raises a potential issue with privacy. Ideally, a unifying record for each student would be something that keeps personal details to a minimum, like a number rather than their full name or email. This is because when working with human subjects, researchers need to ensure they are doing everything they can to protect the private records. Trace data is private information as it could be incredibly descriptive about when, where, or how a student worked. Trace data might also be associated with a grade, which is

also private information. Storing this long-term on a database could leave such information vulnerable for data breaches. Logistically, retaining private information also adds a barrier to entry for researchers interested in examining the data, since access requires authorization from an IRB [100]. Thus, keeping records anonymous whenever possible is a priority.

However, fully anonymizing the data will not work either, currently. At time of writing, each technology supports only their own internal ID meaning identifying information (names, emails, school-given student IDs) is necessary to associate records together. Therefore this information needs to be kept available to help integrate new records into the database to ensure that new activity is properly linked to the correct student. Thus, associating data across platforms is both a difficult undertaking all on its own and one that will require careful consideration to ensure the privacy and safety of all students.

6.4 Data Standardization

Data standardization is another critical step that needs to be addressed in order to create a centralized database for traces from different sources.

As described in previous sections, when different educational software systems each log data in unique ways, integrating records together becomes a major issue. If nothing else, all of the problems laid out in Section 6.2 and Section 6.3 could be ameliorated if student records from Piazza, OpenDSA, Web-CAT and Canvas were already using a standardized ID system for students. More generally, standardization would allow data to take an expected format, meaning integration between systems or with central hubs (like an LMS or centralized data trace database) would be significantly easier. At time of writing, there are no such standardization formats that have been commonly adopted.

Fortunately, there have been significant gains in data standardization that leave me hopeful this challenge can, perhaps, be solved soon. One of the most notable at present is the LTI Standard [19], which has allowed LMSs to start acting like a centralized hub of disparate tools, even if the focus is more on reporting grades back than centralizing data trace collection. For standardizing data traces themselves, the data standardization community has spent years developing the Caliper standard [69] which creates a formalized structure for capturing learning activity data such as search activity, annotations, and forum activity. xAPI [161] is a similar attempt to standardize learning experience data for LA applications, seeking to create a unified format for LMSs, virtual reality, and sensors in a lab. The SPLICE project [141] seeks to support the development of standards and infrastructure for CS education data interchange. Specifically, SPLICE supported development of some standards directly related to specific data types such as ProgSnap and PEML. ProgSnap2 [122] captures snapshots of code, meant to trace the development of student solutions primarily in small programming exercises. PEML [96] attempts to provide a standard for defining small programming exercises with extensions to variations like Parsons problems. Standards like these are beginning to emerge and see use throughout the field. However, while ProgSnap2 and PEML have seen some use within the CER community, Caliper and xAPI are still gaining traction. Thus, while progress is being made, the research community would benefit from a coordinated effort to adopt these standards.

Koedinger et al.'s LearnSphere [70] infrastructure appears to be a highly promicing step in the right direction. This system used the DataShop [149] XML format and suite of analysis tools to store data from different sources across a variety of different classes. LearnSphere, acts as a centralized repository of public data educational data to support interdisciplinary research and help integrate records from a variety of different systems with different types and frequencies of data. These systems, have therefore already, aggregated data from a

variety of sources and anonymized it for public consumption meaning it has solved two important challenges laid out in this chapter.

Additionally, implementing a set of standards for data traces collected from all of these sources in this way also holds tremendous promise for advancing the field overall. Thus far in this paper, we have been considering what it is possible for a single research team to accomplish within (at best) a small set of classes. If multiple research teams collected data, it would allow us to expand our understanding of what a "wider perspective" even means. Standardizing what data looks like when collected from these sources means that research teams can share and integrate datasets—allowing for further advancement in what we can accomplish as a field. With a larger dataset that captures behavior from students across a variety of class contexts, we could begin to see common patterns of behavior among all students. Thus, adopting such standards as those discussed in this section not only facilitates knowledge sharing, but also cultivates a stronger research community.

Chapter 7

Conclusions And Future Work

7.1 Summary of Contributions

This dissertation seeks to answer the following two research questions:

- **RQ1:** What self-regulation skills should be prioritized for intervention within a CS classroom?
- **RQ2:** How do we measure those skills most effectively?

To answer Research Question 1, I conducted two research projects. I (and my co-authors) first assembled a Delphi Process panel of post-secondary CS educators and identified 14 SRL skills that CS students needed to succeed (Chapter 3). Then, I (and my co-authors) conducted a systematic review to assess what the CER community at large considers important and identified 11 skills (Chapter 4). Ultimately, I settled on a finalized list of 12, summarized in Table 5.1.

To answer Research Question 2, I then systematically identified a set of data traces that could best identify those 12 self-regulation skills through interactions with digital software systems (Chapter 5). This approach was, at all times, grounded in existing SRL theory and the recommendations of existing LA research. In many cases, sources for those data traces already exist in the form of common digital education software systems within CS.

Additionally, CS classes that opt to make use of traces from this variety of sources stand to further the study of self regulation through traces significantly, as it is common for current research to only use a single piece of technology.

This dissertation contributes a number of findings to the fields of Computing Education Research and Learning Analytics.

7.1.1 A Shortlist of Skills

Previous decades of research have repeatedly shown how self-regulatory skills are critical for academic success. Even at the post-secondary level, studies show novice programmers proficiency to self-regulate is "...shallow, but trainable.." [83]. Thus, it is not only the responsibility of an educator to teach course content, but to model and help students better understand how to learn.

However, SRL is also a large field and covers skills a student may need to use before, during, and after work. Which skills should be prioritized for targeted instruction? My ultimate answer to that question is simultaneously quite vague and specific.

The vague answer is that it likely depends on the context and culture. The SRL skills that help a student succeed in a typical Introduction to Java course are going to be different from the skills they need for a typical intermediate Data Structures and Algorithms course. These courses likely demand different SRL skills from a student than,the graduate students in a master's-level Software Engineering course. Plus, any of the needs for these three classes at best describe generalized patterns of students within a given class and context. Individual students will, to some extent, need their own unique blend of self-regulatory skills to succeed.

However, such an answer does not allow for an actionable response. It does not give an educator, who may not have an extensive background in SRL theory, any idea of what

SRL skills a class may really need help with. Instructors teaching large classes also gain nothing from the 'it depends' answer. They need a way to assess the needs of 300 students and identify those that need help the most. The study of SRL covers a wide variety of different skills and the current state of the art is still somewhat ambiguous. As discussed in Section 2.1.1, the research community has not solidly decided whether the ideas explored in this dissertation should be considered 'self regulation' or 'metacognition'. If the community has not yet determined the standardized vocabulary to discuss these ideas, asking an educator to implicitly know which self-regulation skills are important to target in their classes is unreasonable.

Thus, I defined a list of 12 highly promising self-regulation skills that are grounded in existing theoretical models of self-regulated learning. These skills are summarized in Table 5.1 and are valuable to both educators and researchers. Both groups now have access to a list of skills that have a strong relationship with theory, and have a strong relationship with the priorities of other CS educators, and are backed by existing CS education research. This means future work can start with skills that already appear to be highly promising, which can help them determine which skills they should prioritize within their own classes. Furthermore, as demonstrated in the systematic review summarized in Chapter 4, some of these skills already have a validated relationship with success in the CER community, while several others seem to be good directions for future work to explore.

These promising skills are especially valuable as my review of the literature also showed that the CER community may benefit from diversifying what skills we study and the methods, we use to capture those skills. Therefore, future research may find it valuable to strike out in a new direction and examine a less-validated, but still valuable, skill.

7.1.2 Empirical Measures for SRL Skills

Measurement is challenging when studying SRL because it is highly internal. This is made even more complicated in the context of post-secondary CS as so much of a student's learning happens as they engage with their coursework outside of class. Additionally, many common approaches to measuring SRL within CER struggle to capture authentic learning behaviors.

Yet measurement is also critical in order to help students learn how to use these skills. Without iterative cycles of practice and formative feedback, a student cannot develop the skills they need to succeed. Furthermore, researchers need effective ways of knowing if an intervention had the desired effect on student behavior.

To that end, I identified a series of data traces that can help future researchers empirically and unobtrusively measure the self-regulatory process in action. Table 5.5, Table 5.6, and Table 5.7 summarize these findings. Unlike the other common forms of measurement, surveys and Think-Aloud studies, data traces are capable of capturing authentic learning through the lens of what students do (rather than what they believe). Following the recommendations of existing LA studies, I took a theory-first approach to identifying traces.

7.2 Future Work

As discussed in Chapter 6, validation is likely the next immediate step for this research. This validation needs to happen at two levels. First, future work will need to validate that the data traces identified in Chapter 5 do indeed allow for inferences to SRL skills. Again, SRL is highly cognitive, meaning that this form of validation will likely require examining student academic outcomes, changes in motivation and self-efficacy, and, potentially, a student self-report inventory to triangulate whether or not these traces capture the skills we care about.

Second, future work will need to validate that these skills are indeed important for academic success in CS. This will likely take the form of crafting various interventions and determining if improving at a specific SRL skill improves academic outcomes.

7.2.1 A Short-Term View

Of all of the skills discussed in this dissertation, Scheduling is perhaps best skill to start with for this form of validation. This is for a number of reasons. First, data traces required to track Scheduling are relatively simple, meaning they will be easier to aggregate into a centralized repository. Simpler data types could also make for easier analysis, require less cleaning and normalizing, and allow researchers to spend more time grappling with the challenges laid out in the previous chapter. Third, Scheduling was one of two skills that currently already has an established connection to academic success. This provides a ground truth for researchers. If we already know that Scheduling as a skill carries a link with academic success, we know that our data traces should produce a similar result. If future studies do *not* find a connection between data traces and academic success, it therefore likely has more to do with the inference from trace to behavior than the inference from behavior to skill.

Additionally, such a study is already highly feasible at Virginia Tech. There are three required courses throughout within the CS major that all typically make use of the same digital ecosystem: CS1114, CS2114, and CS3114. These courses use Canvas as an LMS and typically store assignment instructions using the platform's built-in content hosting features. They also tend to use Web-CAT as an AAT, Piazza as a discussion board platform, and OpenDSA as an e-textbook. This means that once timestamps are aggregated from these data sources for one of these classes, it would be trivial to collect data from the other two

courses as well. All three classes typically assign long-term projects, with at least a week between an assignment's start and end date, meaning it should be clear which students are starting well before a deadline and which students are waiting until the last minute to turn their work in.

Such work would be novel and at a scope that most SRL studies could not achieve. By scope, I mean that this study could examine a much larger population of students over a much longer period of time. At time of writing, the largest-scale interventions I have read about could examine student behavior over the course of a semester [40], going as far as to predict their behavior in a subsequent semester [54]. Many studies opted for a much smaller scale; examining how students behaved as they completed a single assignment or quiz [24, 81, 117]. In the sort of experiment I'm proposing, student Scheduling behavior could be tracked over the course of a semester at minimum. It would be entirely possible to watch that student's time-management abilities develop throughout their career as a college student as they progressed from CS1114 to CS3114.

Following Scheduling, I believe the next highest priority SRL skill from my list to investigate is Knowing When to Seek Help. This is because our Delphi Panel unanimously agreed that Knowing When to Seek Help is a critical skill, but it appears to be neglected from the current state of the art given our findings in the systematic review. Thus, Knowing When to Seek Help is likely important and not yet sufficiently understood. It works well as a next step after studying Scheduling because it also primarily relies on timestamp data. Thus, once the infrastructure for Scheduling was in place, it would not be difficult to start examining this form of Help-Seeking.

7.2.2 A Long-Term View:

Ultimately, future work will eventually need to validate all 12 of these skills and all of the data traces derived in this dissertation. Once all of the traces detailed in Chapter 5 were integrated into a system, I believe the next focus of study should be on how these skills interrelate. Focusing in on a single skill (or a small subset of skills) makes the task of aggregating all of this data together seem more tractable. This requires viewing skills as completely independent of one another. For example, a student could be a strong planner, but not know how to effectively reflect upon their work. Yet this is not always the case. One's proficiency with allocate time to work likely has some relationship with one's proficiency with monitor progress and know when to seek help. Aggregating data from all of these sources together could allow us to see such relationships in action. It could be that this helps future researchers and educators further prioritize what SRL skills to intervene on in their classes. For instance, if improving Task Analysis also dramatically improved four other skills, teaching that skill could end up being more beneficial than teaching a skill that had no such secondary effects. Ultimately, almost nothing is known about how these skills connect to one another within CS at this point, so anything could be possible.

7.3 A Note on Ethics

Within this section, we explore a few matters of ethics raised by this work. The exploration of ethics within the fields of CER and LA are two separate, massive domains. An exhaustive review of all ethical considerations is beyond the scope of this dissertation. Thus, this section will serve as a small start to a few important questions.

Slade and Prinsloo make the following important point within their paper on Learning

Analytics:

> At some point, all institutions supporting student learning must decide what their main purpose really is: to maximize the number of students reaching graduation, to improve the completion rates of students who may be regarded as disadvantaged in some way, or perhaps to simply maximize profits...Amid the emphasis on the role of data and analyses for reporting on student success, retention, and throughput, it is crucial to remember that learning analytics has huge potential to primarily serve *learning* [137].

I believe this is perhaps the most central ethical concern for future researchers to consider.

It is easy to view all of this data as separate from the actual students. As a computer scientist, I can easily frame the teaching of SRL as an optimization problem: how do we maximize surveillance to best increase retention and passing grades? At best, the only ethics in such a framework would be utilitarian with a goal of maximizing good for the most people. Abstracted this way, it is easy to view students as just a resource that we are using to gather data. While this is an advantage of my work, especially for instructors of large classes, teaching SRL skills is about more than maximizing the number of students who pass or achieve a specific grade.

At no point do I wish to imply that the human elements of these issues can or should be overlooked. As an educator, I view my role as a mentor or guide, rather than a repository of knowledge. My job, and the ultimate real goal of this research, should always be centered on helping students learn how they learn best.

With respect to data collection, my goal in this dissertation is to present a variety of data traces that serve as a menu to choose from, not a checklist of requirements to study self regulation as effectively as possible. This work, especially Chapter 5 spends a great deal of time discussing the various ways that it is possible to track student data. However, I do not want to imply that my goal is to extract every possible metric out of our students in the process. For one thing, focusing on optimizing traces at the expense of everything

else encourages unethical behavior. All future research requires the informed consent of the study population and for those performing the research to respect the privacy of students. For another, extracting every possible metric is unnecessary. It is entirely possible to gather data that allows for good inferences using only a subset of the traces, sources, and skills discussed here. Many of the systems we discuss overlap in terms of what kinds of data they collect. For example, practice exercises are sometimes hosted on devoted platforms or could be integrated into an e-textbook. Even with this more diverse perspective, we will have unknowns about what a student is doing. Therefore, we see the use of every possible data trace as a horizon more than an attainable goal.

Once we gather the data, there are also several important things future research must consider in how that data is used. Again, I encourage future researchers to keep the human element in mind here. It is critical to remember that data traces are, at best, inferences. We as researchers are making educated guesses about a whole human being using a series of interactions with a set of digital tools. As educators and researchers, it is our job to make sure we are not prioritizing the data collected over the human being behind that data. While briefly touched upon in Section 2.2, data traces are another step divorced from directly observing student behavior. Even a robust set of data traces cannot and should not be used to assume the full social and cultural context of a student's life. Such context should always be kept in the forefront of any decision-maker's mind when acting upon trace-based observations.

These are only two major considerations out of many that future research will need to grapple with. Questions of what privacy really is for students, the nature of data ownership and the ability to retract collected data from study, and the role of major for-profit companies and their relationship with informed consent (as opposed to researchers) are all important ethical considerations to keep in mind when using trace-based analytics for SRL and do not

yet have definitive answers. For a deeper exploration of these issues, I recommend Hakimi et al.'s 2021 systematic review on the ethical discussions surrounding data traces as a good place to start [53].

7.4 Final Remarks

Research into self regulation is at an exciting point in history. While researchers have been exploring the area for decades, there is both a firm basis of things we do understand and so many unknowns still left to explore. This means that self regulation currently offers a great deal of space for creativity and new ideas in studying student learning within CS.

At this point, it is my assessment that more data and more studies can only serve to benefit the research community as they better come to know this aspect of the student learning experience. There are still so many unknowns with this construct to explore and better understand, still so much to learn. Even research that does not provide conclusive data could help clarify what does and does not work about our existing ideas on this subject. It is my hope that within this dissertation, I have offered some promising directions for that future research.

Thus, with many reasons to try something new and a lot of new tools to try, it is a particularly exciting time to study self regulation as we learn how to teach the act of learning.

Appendices

Appendix A

Additional Tables from the Delphi Process Study

A.1 Informal codes after Phase 1

Name of code	**Code definition**
Scheduling	Allotting time and spacing work out over many sessions
Starting Early	Giving oneself as much time to work with as possible
Understanding the Problem	Knowing fully what's being asked
Decomposing	Breaking a task into smaller components. The act of turning the abstract task into concrete steps.
Design Approach	Crafting overall design or algorithm for code. The act of turning the steps of a problem into pseudo code or code
Monitoring Correctness	Making sure your code is doing what is required (you already know what you must do)
Monitoring Progress	A constant process of checking in to ensure a student is meeting internal milestones
Appropriate Help-Seeking (when)	Balancing struggle against not wasting time before seeking help
Appropriate Help-Seeking (how)	Asking questions that further learning (not "what do I do it's broken")
Communicating with Peers/Group Mates	Discussing ideas with peers
Adaptation	Changing problem solving strategies when something isn't working.
Tinkering/Exploration	Deepening one's understanding by experimenting with code or reading through resources outside of the class (like APIs)
Emotional Regulation	Being able to push through negative emotions to continue coding or able to recover from failure
Reflection	Awareness of yourself and what you're doing (internal and cognitive)
Response to Feedback	Post assignment/test understanding feedback from results and drawing conclusions about what one might need to change going forward.

A.2 Refined skill list after Phase 2

Category	Skill Name	Skill Definition
Planning	Understanding the Problem	Taking steps to ensure understanding of expectations and requirements
	Exploration	Deepening one's understanding through activities that are outside of what is required for credit in a classroom
Execution	Decomposing	The act of turning a large task into smaller steps
	Scheduling	Intentional pre-allocation of time for work
	Starting Early	Giving oneself as much time to work with as possible
Monitoring	Monitoring Progress	A regular process of checking in to ensure you are meeting internal milestones
	Monitoring Correctness	Consistently checking in to make sure your work matches requirements
	Emotional Regulation	Recognizing emotions when they come up and adapting your learning strategies to address them
Response	Working with Peers or Group Mates	Intentionally behaving and communicating in a way that facilitates effective work within a group
	Knowing When to Seek Help	Balancing trying to work through a challenge on your own with spending time productively
	Knowing How to Seek Help	Discerning appropriate methods and channels of acquiring help
	Adaptation	Intentionally trying another strategy after recognizing you are stuck
Reflection	Reflection Using Internal Standards	Critical evaluation of past success, challenges, and opportunities for improvement against one's internal criteria for success
	Reflection Using External Standards	Taking steps to understand external feedback and drawing conclusions about what one needs to change going forward

Appendix B

Additional Tables from the Systematic Review

B.1 Final Corpus

Citation	Title	Authors	Year	Published in
[13]	Pencil Puzzles for Introductory Computer Science: an Experience- and Gender-Neutral Context	Butler, Bezakova, and Fluet	2017	SIGCSE
[144]	Exam Wrappers: Not a Silver Bullet	Stephenson, Craig, Zingaro, Horton, Heap, and Huynh	2017	SIGCSE
[85]	Student Challenges, Strategies, and Learning Within the Data Mine Learning Community	Lyon, Jaiswal, Magana, Gundlach, and Ward	2021	FIE
[32]	Improving Content Learning and Student Perceptions in CS1 with Scrumage	Duvall, Spurlock, Hutchings, and Duvall	2021	SIGCSE
[143]	A Study of the Relationship Between a CS1 Student's Gender and Performance Versus Gauging Understanding and Study Tactics	Stephens-Martinez	2021	SIGCSE
[28]	Patterns of Academic Help-Seeking in Undergraduate Computing Students	Doebling and Kazerouni	2021	Koli Calling
[109]	Computational Creativity Exercises: An Avenue for Promoting Learning in Computer Science	Peteranetz, Flanigan, Shell, and Soh	2017	IEEE Transactions on Education
[73]	Targeting Metacognition by Incorporating Student-Reported Confidence Estimates on Self-Assessment Quizzes	Lee and Liao	2021	SIGCSE
[130]	The Impact of Programming Project Milestones on Procrastination, Project Outcomes, and Course Outcomes: A Quasi-Experimental Study in a Third-Year Data Structures Course	Shaffer and Kazerouni	2021	SIGCSE
[25]	A Closer Look at Metacognitive Scaffolding: Solving Test Cases Before Programming	Denny, Prather, Becker, Albrecht, Loksa, and Pettit	2019	Koli Calling
[26]	Promoting Early Engagement with Programming Assignments Using Scheduled Automated Feedback	Denny, Whalley, Leinonen	2021	ACE
[74]	Does the Early Bird Catch the Worm? Earliness of Students' Work and its Relationship with Course Outcomes	Leinonen, Castro, Hellas	2021	ITiCSE
[35]	Examining Classroom Interventions to Reduce Procrastination	Martin, Edwards, and Shaffer	2015	ITiCSE
[24]	Self-Predicted and Actual Performance in an Introductory Programming Course	Denny, Luxton-Reilly, Hamer, Dahlstrom, and Purchase	2010	ITiCSE
[76]	Using and Collecting Fine-Grained Usage Data to Improve Online Learning Materials	Leppanen, Leinonen, Ihantola, and Hellas	2017	ICSE
[118]	First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts	Prather, Pettit, Becker, Denny, Loksa, Peters, Albrecht, and Masci	2019	SIGCSE
[15]	Understanding the Relationships Between Self-Regulated Learning and Students Source Code in a Computer Programming Course	Castellanos, Restrepo-Calle, González, and Echeverry	2017	FIE
[3]	In Situ Identification of Student Self-Regulated Learning Struggles in Programming Assignments	Arakawa, Hao, Greer, Ding, Hundhausen, and Peterson	2021	SIGCSE
[62]	Supporting Self-Regulated Learning with Visualizations in Online Learning Environments	Ilves, Leinonen, and Hellas	2018	SIGCSE
[121]	Getting by with help from my friends: Group Study in Introductory Programming Understood as Socially Shared Regulation	Prather, Margulieux, Whalley, Denny, Reeves, Becker, Singh, Powell, and Bosch	2022	ICER
[75]	Pauses and Spacing in Learning to Program	Leppänen, Leinonen, and Hellas	2016	Koli Calling
[132]	A Qualitative Study on How Students Interact with Quizzes and Estimate Confidence on Their Answers	Shah, Lee, Barretto, and Liao	2021	ITiCSE
[14]	Factors for Success in Online CS1	Campbell.Horton, and Craig	2016	ITiCSE
[81]	The Role of Self-Regulation in Programming Problem Solving Process and Success	Loksa and Ko	2016	ICER

Table B.1: Papers Reviewed for this Work: Part 1

Citation	Title	Authors	Year	Published in
[79]	Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance	Lishinski, Yadav, Good, and Enbody	2016	ICER
[125]	What Help Do Students Seek in TA Office Hours?	Ren, Krishnamurthi, and Fisler	2019	ICER
[139]	Applying Computational Analysis of Novice Learners' Computer Programming Patterns to Reveal Self-Regulated Learning, Computational Thinking, a nd Learning Performance	Song, Hong, and Oh	2021	Computers in Human Behavior
[151]	Do Students Know What They Think They Know? Assessing Student Confidence in a Computer Graphics Course	Urness	2016	Journal of Computing Sciences in Colleges
[18]	Investigating Patterns of Study Persistence on Self-Assessment Platform of Programming Problem-Solving	Chung and Hsiao	2020	SIGCSE
[58]	Comparison of Grade Replacement and Weighted Averages for Second-Chance Exams	Herman, Cai, Bretl, Zilles, and West	2020	ICER
[117]	Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools	Prather, Pettit, McMurry, Peters, Homer, and Cohen	2018	ICER
[162]	Exploring the Impact of Voluntary Practice and Procrastination in an Introductory Programming Course	Zhang, Cunningham, Iyer, Baker, and Fouh	2022	SIGCSE
[103]	Empirical Evidence for the Existence and Uses of Metacognition in Computer Science Problem Solving	Parham, Gugerty, and Stevenson	2010	SIGCSE
[107]	Struggling To Keep Tabs on Capstone Projects: A Chatbot to Tackle Student Procrastination	Pereira and Díaz	2021	ACM Transactions on Computing Education
[39]	Identifying Computer Science Self-Regulated Learning Strategies	Falkner, Vivian, and Falkner	2014	ITiCSE
[136]	What Do They Note? An Exploratory Investigation Into The Characteristics of CS Students' Notes	Singh, Tempero, Luxton-Reilly, and Zhang	2021	CSERC
[57]	Proof by Incomplete Enumeration and Other Logical Misconceptions	Herman, Kaczmarczyk, Loui, and Zilles	2008	ICER
[145]	Expression of Abstraction: Self explanation in Code Production	Sudol-DeLyser	2015	SIGCSE

Table B.2: Papers Reviewed for this Work

Key for Conference Acronyms in Table B.1 and Table B.2:

- **SIGCSE** - Special Interest Group Computer Science Education Conference
- **FIE** - Frontiers in Education Conference
- **ACE** - Australasian Computing Education Conference
- **ITiCSE** - Innovation and Technology in Computer Science Education Conference
- **ICSE** - International Conference on Software Engineering
- **ICER** - International Computing Education Research Conference
- **CSERC** - Computer Science Education Research Conference

Bibliography

[1] Ali Alharbi, Frans Henskens, and Michael Hannaford. Personalised learning object system based on self-regulated learning theories. *Int. J. Eng. Pedagog.*, 4(3):24–35, 2014.

[2] Eric Araka, Elizaphan Maina, Rhoda Gitonga, and Robert Oboko. Research trends in measurement and intervention tools for self-regulated learning for e-learning environments—systematic review (2008–2018). *Research and Practice in Technology Enhanced Learning*, 15:1–21, 2020.

[3] Kai Arakawa, Qiang Hao, Tyler Greer, Lu Ding, Christopher D. Hundhausen, and Abigayle Peterson. In situ identification of student self-regulated learning struggles in programming assignments. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, page 467–473, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432357.

[4] Kimberly E. Arnold and Matthew D. Pistilli. Course signals at purdue: using learning analytics to increase student success. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, LAK '12, page 267–270, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311113. doi: 10.1145/2330601.2330666. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/2330601.2330666.

[5] Roger Azevedo, Amy Witherspoon, Amber Chauncey, Candice Burkett, and Ashley

Fike. Metatutor: A metacognitive tool for enhancing self-regulated learning. In *2009 AAAI Fall symposium series*, 2009.

[6] Linda Baker and Ann L Brown. Metacognitive skills and reading. technical report no. 188. 1980.

[7] Ryan Shaun Baker, Albert T. Corbett, Kenneth R. Koedinger, and Angela Z. Wagner. Off-task behavior in the cognitive tutor classroom: When students "game the system". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, page 383–390, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581137028. doi: 10.1145/985692.985741. URL https://doi.org/10.1145/985692.985741.

[8] A. Bandura. Social foundations of thought and action. 1986. URL https://api.semanticscholar.org/CorpusID:142519016.

[9] Robert Bodily and Katrien Verbert. Review of research on student-facing learning analytics dashboards and educational recommender systems. *IEEE Transactions on Learning Technologies*, 10(4):405–418, 2017. doi: 10.1109/TLT.2017.2740172.

[10] Monique Boekaerts. Understanding students' affective processes in the classroom. In *Emotion in Education*, pages 37–56. Elsevier, 2007. ISBN 9780123725455. doi: 10.1016/B978-012372545-5/50004-6.

[11] Jaclyn Broadbent and Walter L Poon. Self-regulated learning strategies academic achievement in online higher education learning environments: A systematic review. *The internet and higher education*, 27:1–13, 2015.

[12] Peter Brusilovsky, Ken Koedinger, David A. Joyner, and Thomas W. Price. Building an infrastructure for computer science education research and practice at scale. In

Proceedings of the Seventh ACM Conference on Learning @ Scale, L@S '20, page 211–213, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379519. doi: 10.1145/3386527.3405936. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3386527.3405936.

[13] Zack Butler, Ivona Bezakova, and Kimberly Fluet. 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*, SIGCSE '17, page 93–98, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346986. doi: 10.1145/3017680.3017765.

[14] Jennifer Campbell, Diane Horton, and Michelle Craig. Factors for success in online cs1. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '16, page 320–325, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342315. doi: 10.1145/2899415.2899457.

[15] Hugo Castellanos, Felipe Restrepo-Calle, Fabio A. González, and Jhon Jairo Ramírez Echeverry. Understanding the relationships between self-regulated learning and students source code in a computer programming course. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–9, 2017. doi: 10.1109/FIE.2017.8190467.

[16] Hsi-Min Chen, Wei-Han Chen, Nien-Lin Hsueh, Chi-Chen Lee, and Chia-Hsiu Li. Progedu - an automatic assessment platform for programming courses. In *2017 International Conference on Applied System Innovation (ICASI)*, pages 173–176, 2017. doi: 10.1109/ICASI.2017.7988376.

[17] Zui Cheng, Zhuo Zhang, Qian Xu, Yukiko Maeda, and Peidi Gu. A meta-analysis addressing the relationship between self-regulated learning strategies and academic

performance in online higher education. *Journal of Computing in Higher Education*, pages 1–30, 2023.

[18] Cheng-Yu Chung and I-Han Hsiao. 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*, SIGCSE '20, page 162–168, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367936. doi: 10.1145/3328778.3366827.

[19] 1EdTech Consortium. Learning tools interoperability (lti), 2024. URL https://www.1edtech.org/standards/lti.

[20] Tudor. Cristea, Chris Snijders, Uwe Matzat, and Ad Kleingeld. Unobtrusive measurement of self-regulated learning: A clickstream-based multi-dimensional scale. *Education and Information Technologies*, 2023. doi: https://doi.org/10.1007/s10639-023-12372-6.

[21] Rajdeep Das, Umair Z. Ahmed, Amey Karkare, and Sumit Gulwani. Prutor: A system for tutoring CS1 and collecting student programs for analysis. *CoRR*, abs/1608.03828, 2016. URL http://arxiv.org/abs/1608.03828.

[22] Dan Davis, Guanliang Chen, Ioana Jivet, Claudia Hauff, and Geert-Jan Houben. Encouraging metacognition & self-regulation in moocs through increased learner feedback. In *LAL@ LAK*, pages 17–22, 2016.

[23] Richard A DeMillo, Richard J Lipton, and Frederick G Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41, 1978.

[24] Paul Denny, Andrew Luxton-Reilly, John Hamer, Dana B. Dahlstrom, and Helen C. Purchase. Self-predicted and actual performance in an introductory program-

ming course. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10, page 118–122, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605588209. doi: 10.1145/1822090.1822124.

[25] Paul Denny, James Prather, Brett A. Becker, Zachary Albrecht, Dastyni Loksa, and Raymond Pettit. 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 Calling '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450377157. doi: 10.1145/3364510.3366170.

[26] Paul Denny, Jacqueline Whalley, and Juho Leinonen. Promoting early engagement with programming assignments using scheduled automated feedback. In *Proceedings of the 23rd Australasian Computing Education Conference*, ACE '21, page 88–95, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450389761. doi: 10.1145/3441636.3442309.

[27] Flavia Devonas Hoffmann and Kristine Høeg Karlsen. Choreographic infrastructuring for design things: A new method for participatory design in teacher education. pages 230–240, 2022. doi: 10.1145/3536169.3537796.

[28] Augie Doebling and Ayaan M. Kazerouni. Patterns of academic help-seeking in undergraduate computing students. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*, Koli Calling '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384889. doi: 10.1145/3488042.3488052.

[29] Molly Domino and Clifford Shaffer. Taking a wider perspective: Assessing the advan-

tages of using a wider digital ecosystem to study self-regulated learning skills. *Journal Anonomized For Review*, 20XX.

[30] Molly Domino, Bob Edmison, Stephen Edwards, Rifat Mansur, Alexandra Thompson, and Clifford Shaffer. Identifying critical self-regulated learning skills: A delphi process study. *Computer Science Education*.

[31] Molly Domino, Alexandra Thompson, Alexander Hicks, Alan Jamieson, Khushi Parajuli, Bob Edmison, Stephen Edwards, and Clifford Shaffer. How can we know when we see it? a systematic review of cognitive control skills and behaviors. *ACM Transactions on Computing Education*, 20XX.

[32] Shannon Duvall, Scott Spurlock, Dugald Ralph Hutchings, and Robert C. Duvall. Improving content learning and student perceptions in cs1 with scrumage. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, page 474–480, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432415.

[33] Stephen H. Edwards and Krishnan Panamalai Murali. Codeworkout: Short programming exercises with built-in data collection. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '17, page 188–193, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450347044. doi: 10.1145/3059009.3059055. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3059009.3059055.

[34] Stephen H. Edwards and Manuel A. Perez-Quinones. Web-cat: automatically grading programming assignments. *SIGCSE Bull.*, 40(3):328, jun 2008. ISSN 0097-8418. doi: 10.1145/1597849.1384371. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/1597849.1384371.

[35] Stephen H. Edwards, Joshua Martin, and Clfford A. Shaffer. Examining classroom interventions to reduce procrastination. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, page 254–259, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334402. doi: 10.1145/2729094.2742632.

[36] Anastasia Efklides. Interactions of metacognition with motivation and affect in self-regulated learning: The masrl model. *Educational psychologist*, 46(1):6–25, 2011. doi: 10.1080/00461520.2011.538645.

[37] Barbara J. Ericson and Bradley N. Miller. Runestone: A platform for free, on-line, and interactive ebooks. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 1012–1018, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367936. doi: 10.1145/3328778.3366950. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3328778.3366950.

[38] Barbara J. Ericson, Paul Denny, James Prather, Rodrigo Duran, Arto Hellas, Juho Leinonen, Craig S. Miller, Briana B. Morrison, Janice L. Pearce, and Susan H. Rodger. Parsons problems and beyond: Systematic literature review and empirical study designs. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '22, page 191–234, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9798400700101. doi: 10.1145/3571785.3574127. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3571785.3574127.

[39] Katrina Falkner, Rebecca Vivian, and Nickolas J.G. Falkner. Identifying computer science self-regulated learning strategies. In *Proceedings of the 2014 Conference on Innovation*

Technology in Computer Science Education, ITiCSE '14, page 291–296, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328333. doi: 10.1145/2591708.2591715.

[40] Abraham E. Flanigan, Markeya S. Peteranetz, Duane F. Shell, and Leen-Kiat Soh. Exploring changes in computer science students' implicit theories of intelligence across the semester. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, page 161–168, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336307. doi: 10.1145/2787622.2787722.

[41] Carlton J Fong, Cassandra Gonzales, Christie Hill-Troglin Cox, and Holly B Shinn. Academic help-seeking and achievement of postsecondary students: A meta-analytic investigation. *Journal of Educational Psychology*, 115(1):1, 2023.

[42] Association for Computing Machinery. About the acm digital library, 2023.

[43] Society for Learning Analytics Research (SoLAR). What is learning analytics?, Mar 2021. URL https://www.solaresearch.org/about/what-is-learning-analytics/.

[44] Joy Galaige, Geraldine Torrisi Steele, Sebastian Binnewies, and Kewen Wang. A framework for designing student-facing learning analytics to support self-regulated learning. *IEEE Transactions on Learning Technologies*, 15(3):376–391, 2022. doi: 10.1109/TLT.2022.3176968.

[45] Rita Garcia, Katrina Falkner, and Rebecca Vivian. Systematic literature review: Self-regulated learning strategies using e-learning tools for computer science. *Computers Education*, 123:150–163, 2018. doi: 10.1016/j.compedu.2018.05.006.

[46] Dragan Gašević, Shane Dawson, Tim Rogers, and Danijela Gasevic. Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success. *The Internet and Higher Education*, 28:68–84, 2016.

[47] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey Herman, Lisa Kaczmarczyk, Michael C. Loui, and Craig Zilles. Identifying important and difficult concepts in introductory computing courses using a delphi process. *SIGCSE Bull.*, 40(1):256–260, mar 2008. ISSN 0097-8418. doi: 10.1145/1352322.1352226.

[48] Geraldine Gray and Yoav Bergner. A practitioner's guide to measurement in learning analytics: Decisions, opportunities, and challenges. *Handbook of learning analytics,*, pages 20–28, 2022.

[49] Jeffrey A Greene, Leigh Anna Hutchison, Lara-Jeane Costa, and Helen Crompton. Investigating how college students' task definitions and plans relate to self-regulated learning processing and understanding of a complex science topic. *Contemporary Educational Psychology*, 37(4):307–320, 2012. doi: https://doi.org/10.1016/j.cedpsych.2012.02.002.

[50] Jeffrey Alan Greene and Roger Azevedo. A theoretical review of winne and hadwin's model of self-regulated learning: New perspectives and directions. *Review of educational research*, 77(3):334–372, 2007.

[51] AF Hadwin, S Järvelä, and M Miller. Self-regulated, co-regulated, and socially shared regulation of learning. 2011.

[52] Allyson Hadwin, Sanna Järvelä, and Mariel Miller. Self-regulation, co-regulation, and shared regulation in collaborative learning environments. In *Handbook of Self-Regulation of Learning and Performance*, pages 83–106. Routledge, 2017. ISBN 9781315697048.

[53] Laura Hakimi, Rebecca Eynon, and Victoria A Murphy. The ethics of using digital trace data in education: A thematic review of the research landscape. *Review of educational research*, 91(5):671–717, 2021.

[54] Joonas Häkkinen, Petri Ihantola, Matti Luukkainen, Antti Leinonen, and Juho Leinonen. Persistence of time management behavior of students and its relationship with performance in software projects. In *Proceedings of the 17th ACM Conference on International Computing Education Research*, ICER 2021, page 92–100, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383264. doi: 10.1145/3446871.3469767. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3446871.3469767.

[55] Brian Harrington, Shichong Peng, Xiaomeng Jin, and Minhaz Khan. Gender, confidence, and mark prediction in cs examinations. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE 2018, page 230–235, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357074. doi: 10.1145/3197091.3197116.

[56] Julian T Hart. Memory and the feeling-of-knowing experience. *Journal of educational psychology*, 56(4):208, 1965.

[57] Geoffrey L. Herman, Lisa Kaczmarczyk, Michael C. Loui, and Craig Zilles. Proof by incomplete enumeration and other logical misconceptions. In *Proceedings of the Fourth International Workshop on Computing Education Research*, ICER '08, page 59–70, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582160. doi: 10.1145/1404520.1404527.

[58] Geoffrey L. Herman, Zhouxiang Cai, Timothy Bretl, Craig Zilles, and Matthew West. Comparison of grade replacement and weighted averages for second-chance exams. In

Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER '20, page 56–66, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370929. doi: 10.1145/3372782.3406260.

[59] Chun-Heng Ho. Some phenomena of problem decomposition strategy for design thinking: Differences between novices and experts. *Design Studies*, 22(1):27–45, 2001.

[60] David Hovemeyer and Jaime Spacco. Cloudcoder: a web-based programming exercise system. *J. Comput. Sci. Coll.*, 28(3):30, jan 2013. ISSN 1937-4771.

[61] I-Han Hsiao, Po-Kai Huang, and Hannah Murphy. Uncovering reviewing and reflecting behaviors from paper-based formal assessment. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, LAK '17, page 319–328, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348706. doi: 10.1145/3027385.3027415. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3027385.3027415.

[62] Kalle Ilves, Juho Leinonen, and Arto Hellas. Supporting self-regulated learning with visualizations in online learning environments. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 257–262, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351034. doi: 10.1145/3159450.3159509.

[63] Michael S. Irwin and Stephen H. Edwards. Can mobile gaming psychology be used to improve time management on programming assignments? In *Proceedings of the ACM Conference on Global Computing Education*, CompEd '19, page 208–214, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362597. doi: 10.1145/3300115.3309517. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3300115.3309517.

[64] P.M. Johnson, Hongbing Kou, J.M. Agustin, Qin Zhang, A. Kagawa, and T. Yamashita. Practical automated process and product metric collection and analysis in a classroom setting: lessons learned from hackystat-uh. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.*, pages 136–144, 2004. doi: 10.1109/ISESE.2004.1334901.

[65] Rose Catherine Kanjirathinkal, Rashmi Gangadharaiah, Karthik Visweswariah, and Dinesh Raghu. Semi-supervised answer extraction from discussion forums. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1–9, 2013.

[66] Oscar Karnalim and Simon. Promoting code quality via automated feedback on student submissions. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, 2021. doi: 10.1109/FIE49875.2021.9637193.

[67] Ayaan M. Kazerouni, Stephen H. Edwards, T. Simin Hall, and Clifford A. Shaffer. Deveventtracker: Tracking development events to assess incremental development and procrastination. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '17, page 104–109, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450347044. doi: 10.1145/3059009.3059050. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3059009.3059050.

[68] Ayaan M. Kazerouni, Stephen H. Edwards, and Clifford A. Shaffer. Quantifying incremental development practices and their relationship to procrastination. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, ICER '17, page 191–199, New York, NY, USA, 2017. Association for Com-

puting Machinery. ISBN 9781450349680. doi: 10.1145/3105726.3106180. URL https://doi.org/10.1145/3105726.3106180.

[69] Yeon Hee Kim and Jin-Ho Ahn. A study on the application of big data to the korean college education system. *Procedia Computer Science*, 91:855–861, 2016.

[70] Ken Koedinger, Ran Liu, John Stamper, Candace Thille, and Phil Pavlik. Community based educational data repositories and analysis tools. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, LAK '17, page 524–525, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348706. doi: 10.1145/3027385.3029442. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3027385.3029442.

[71] Ari Korhonen and Lauri Malmi. Algorithm simulation with automatic assessment. In *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSEconference on Innovation and Technology in Computer Science Education*, ITiCSE '00, page 160–163, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581132077. doi: 10.1145/343048.343157. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/343048.343157.

[72] Sophia Krause-Levy, William G. Griswold, Leo Porter, and Christine Alvarado. The relationship between sense of belonging and student outcomes in cs1 and beyond. In *Proceedings of the 17th ACM Conference on International Computing Education Research*, ICER 2021, page 29–41, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383264. doi: 10.1145/3446871.3469748.

[73] Priscilla Lee and Soohyun Nam Liao. Targeting metacognition by incorporating student-reported confidence estimates on self-assessment quizzes. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21,

page 431–437, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432377.

[74] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. Does the early bird catch the worm? earliness of students' work and its relationship with course outcomes. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE '21, page 373–379, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382144. doi: 10.1145/3430665.3456383.

[75] Leo Leppänen, Juho Leinonen, and Arto Hellas. Pauses and spacing in learning to program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, page 41–50, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450347709. doi: 10.1145/2999541.2999549.

[76] Leo Leppanen, Juho Leinonen, Petri Ihantola, and Arto Hellas. Using and collecting fine-grained usage data to improve online learning materials. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 4–12, 2017. doi: 10.1109/ICSE-SEET.2017.12.

[77] Kathryn Ley and Dawn B Young. Instructional principles for self-regulation. *Educational Technology Research and Development*, 49(2):93–103, 2001.

[78] Soohyun Nam Liao, Kartik Shah, William G. Griswold, and Leo Porter. A quantitative analysis of study habits among lower- and higher-performing students in cs1. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE '21, page 366–372, 2021. ISBN 9781450382144. doi: 10.1145/3430665.3456350.

[79] Alex Lishinski, Aman Yadav, Jon Good, and Richard Enbody. 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*, ICER '16, page 211–220, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450344494. doi: 10.1145/2960310.2960329.

[80] Canvas LMS. Canvas by instructure: World's 1 teaching and learning software. URL https://www.instructure.com/canvas/.

[81] Dastyni Loksa and Amy J. Ko. The role of self-regulation in programming problem solving process and success. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, ICER '16, page 83–91, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450344494. doi: 10.1145/2960310.2960334.

[82] Dastyni Loksa, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. Programming, problem solving, and self-awareness: Effects of explicit guidance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 1449–1461, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858252.

[83] Dastyni Loksa, Benjamin Xie, Harrison Kwik, and Amy J. Ko. *Investigating Novices' In Situ Reflections On Their Programming Process*, page 149–155. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450367936. doi: 10.1145/3328778.3366846.

[84] Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. Metacognition and self-regulation in programming

education: Theories and exemplars of use. *ACM Trans. Comput. Educ.*, 22(4), sep 2022. doi: 10.1145/3487050.

[85] Joseph A. Lyon, Aparajita Jaiswal, Alejandra J. Magana, Ellen Gundlach, and Mark Daniel Ward. Student challenges, strategies, and learning within the data mine learning community. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–7, 2021. doi: 10.1109/FIE49875.2021.9637437.

[86] Marko Lüftenegger, Barbara Schober, Rens Schoot, Petra Wagner, Monika Finsterwald, and Christiane Spiel. Lifelong learning as a goal - do autonomy and self-regulation in school result in well prepared pupils? *Learning and Instruction - LEARN INSTR*, 22:27–36, 02 2012. doi: 10.1016/j.learninstruc.2011.06.001.

[87] Therese H Macan, Comila Shahani, Robert L Dipboye, and Amanda P Phillips. College students' time management: Correlations with academic performance and stress. *Journal of educational psychology*, 82(4):760, 1990.

[88] Jonna Malmberg, Eetu Haataja, and Sanna Järvelä. Exploring the connection between task difficulty, task perceptions, physiological arousal and learning outcomes in collaborative learning situations. *Metacognition and Learning*, 17(3):793–811, 2022. doi: 10.1007/s11409-022-09320-z.

[89] Catherine Marshall and Gretchen B Rossman. Managing, analyzing, and interpreting data. In *Designing Qualitative Research*, pages 217–225. Sage Publications, 2014. ISBN 9781452271002.

[90] Joshua Martin, Stephen H. Edwards, and Clfford A. Shaffer. The effects of procrastination interventions on programming project success. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, page 3–11, 2015. ISBN 9781450336307. doi: 10.1145/2787622.2787730.

[91] Roberto Martinez-Maldonado, Abelardo Pardo, Negin Mirriahi, Kalina Yacef, Judy Kay, and Andrew Clayphan. Latux: An iterative workflow for designing, validating, and deploying learning analytics visualizations. *Journal of Learning Analytics*, 2(3): 9–39, 2015.

[92] Samiha Marwan, Anay Dombe, and Thomas W. Price. 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*, ITiCSE '20, page 54–60, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368742. doi: 10.1145/3341525.3387394.

[93] Wannisa Matcha, Nora'ayu Ahmad Uzir, Dragan Gašević, and Abelardo Pardo. A systematic review of empirical studies on learning analytics dashboards: A self-regulated learning perspective. *IEEE Transactions on Learning Technologies*, 13(2):226–245, 2020. doi: 10.1109/TLT.2019.2916802.

[94] Igor Mekterović, Ljiljana Brkić, Boris Milašinović, and Mirta Baranović. Building a comprehensive automated programming assessment system. *IEEE Access*, 8:81154–81172, 2020. doi: 10.1109/ACCESS.2020.2990980.

[95] Marcus Messer, Neil C. C. Brown, Michael Kölling, and Miaojing Shi. Automated grading and feedback tools for programming education: A systematic review. *ACM Trans. Comput. Educ.*, 24(1), feb 2024. doi: 10.1145/3636515. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3636515.

[96] Divyansh S. Mishra and Stephen H. Edwards. The programming exercise markup language: Towards reducing the effort needed to use automated grading tools. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, page 395–401, New York, NY, USA, 2023. Association for Computing

Machinery. ISBN 9781450394314. doi: 10.1145/3545945.3569734. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3545945.3569734.

[97] Moodle. Moodle. URL https://moodle.org/.

[98] Victoria Naisola-Ruiter. The delphi technique: a tutorial. *Research in Hospitality Management*, 12(1):91–97, 2022.

[99] Marlen Niederberger and Julia Spranger. Delphi technique in health sciences: a map. *Frontiers in public health*, 8:457, 2020.

[100] University of Rhode Island. Human subjects protections: Does my research need irb review?, 2024. URL https://web.uri.edu/research-admin/office-of-research-integrity/human-subjects-protections/does-my-research-need-irb-review/.

[101] Ernesto Panadero. A review of self-regulated learning: Six models and four directions for research. *Frontiers in psychology*, page 422, 2017. doi: 10.3389/fpsyg.2017.00422. URL https://www.frontiersin.org/articles/10.3389/fpsyg.2017.00422/full.

[102] Ernesto Panadero, Julia Klug, and Sanna Järvelä. Third wave of measurement in the self-regulated learning field: When measurement and intervention come hand in hand. *Scandinavian Journal of Educational Research*, 60(6):723–735, 2016.

[103] Jennifer Parham, Leo Gugerty, and D. E. Stevenson. 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*, SIGCSE '10, page 416–420, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300063. doi: 10.1145/1734263.1734406.

[104] Nick Parlante. Codingbat code practice, 2017. URL http://codingbat.com/.

[105] Martin Pärtel, Matti Luukkainen, Arto Vihavainen, and Thomas Vikberg. Test my code. *Int. J. Technol. Enhanc. Learn.*, 5(3/4):271–283, feb 2013. ISSN 1753-5255. doi: 10.1504/IJTEL.2013.059495. URL https://doi-org.ezproxy.lib.vt.edu/10.1504/IJTEL.2013.059495.

[106] Eric J Paulson and Laurie Bauer. Goal setting as an explicit element of metacognitive reading and study strategies for college reading. *NADE Digest*, 5(3):41–49, 2011.

[107] Juanan Pereira and Óscar Díaz. Struggling to keep tabs on capstone projects: A chatbot to tackle student procrastination. *ACM Trans. Comput. Educ.*, 22(1), oct 2021. doi: 10.1145/3469127.

[108] Nancy E Perry and Philip Winne. Learning from learning kits: gstudy traces of students' self-regulated engagements with computerized content. *Educational Psychology Review*, 18(3):211–228, 2006.

[109] Markeya S. Peteranetz, Abraham E. Flanigan, Duane F. Shell, and Leen-Kiat Soh. Computational creativity exercises: An avenue for promoting learning in computer science. *IEEE Transactions on Education*, 60(4):305–313, 2017. doi: 10.1109/TE.2017.2705152.

[110] Paul R Pintrich and Elisabeth V De Groot. Motivational and self-regulated learning components of classroom academic performance. *Journal of educational psychology*, 82 (1):33, 1990. doi: 10.1037/0022-0663.82.1.33.

[111] Paul R Pintrich and Elisabeth V De Groot. Motivational and self-regulated learning components of classroom academic performance. *Journal of educational psychology*, 82 (1):33, 1990.

[112] Paul R Pintrich, David AF Smith, Teresa Garcia, and Wilbert J McKeachie. Reliability and predictive validity of the motivated strategies for learning questionnaire (mslq). *Educational and psychological measurement*, 53(3):801–813, 1993. doi: 10.1177/0013164493053003024.

[113] Paul R Pintrich, Christopher A Wolters, and Gail P Baxter. Assessing metacognition and self-regulated learning. In *Issues in the Measurement of Metacognition*, pages 43–97, 2000.

[114] Paul R Pintrich et al. A manual for the use of the motivated strategies for learning questionnaire (mslq). 1991. URL https://files.eric.ed.gov/fulltext/ED338122.pdf.

[115] Annette Plüddemann, Jeffrey K Aronson, Igho Onakpoya, Carl Heneghan, and Kamal R Mahtani. Redefining rapid reviews: A flexible framework for restricted systematic reviews. *BMJ Evidence-Based Medicine*, 23(6):201–203, 2018. ISSN 2515-446X. doi: 10.1136/bmjebm-2018-110990.

[116] Pierre Pluye, Quan Nha Hong, and Isabelle Vedel. Toolkit for mixed studies reviews (v3), 2016.

[117] James Prather, Raymond Pettit, Kayla McMurry, Alani Peters, John Homer, and Maxine Cohen. Metacognitive difficulties faced by novice programmers in automated assessment tools. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, ICER '18, page 41–50, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356282. doi: 10.1145/3230977.3230981.

[118] James Prather, Raymond Pettit, Brett A. Becker, Paul Denny, Dastyni Loksa, Alani Peters, Zachary Albrecht, and Krista Masci. First things first: Providing metacognitive

scaffolding for interpreting problem prompts. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, page 531–537, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450358903. doi: 10.1145/3287324.3287374.

[119] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. 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*, ICER '20, page 2–13, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370929. doi: 10.1145/3372782.3406263.

[120] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. 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*, ICER '20, page 2–13, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370929. doi: 10.1145/3372782.3406263. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3372782.3406263.

[121] James Prather, Lauren Margulieux, Jacqueline Whalley, Paul Denny, Brent N. Reeves, Brett A. Becker, Paramvir Singh, Garrett Powell, and Nigel Bosch. Getting by with help from my friends: Group study in introductory programming understood as socially shared regulation. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*, ICER '22, page 164–176, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391948. doi: 10.1145/3501385.3543970.

[122] Thomas W. Price, David Hovemeyer, Kelly Rivers, Ge Gao, Austin Cory Bart, Ayaan M. Kazerouni, Brett A. Becker, Andrew Petersen, Luke Gusukuma, Stephen H. Edwards, and David Babcock. Progsnap2: A flexible format for programming process data. ITiCSE '20, page 356–362, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368742. doi: 10.1145/3341525.3387373. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3341525.3387373.

[123] Minna Puustinen and Lea Pulkkinen. Models of self-regulated learning: A review. *Scandinavian journal of educational research*, 45(3):269–286, 2001.

[124] Jhon Jairo Ramírez Echeverry, Àgueda García Carrillo, and Fredy Andres Olarte Dussan. Adaptation and validation of the motivated strategies for learning questionnaire-mslq-in engineering students in colombia. *International journal of engineering education*, 32(4):1774–1787, 2016.

[125] Yanyan Ren, Shriram Krishnamurthi, and Kathi Fisler. What help do students seek in ta office hours? In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 41–49, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361859. doi: 10.1145/3291279.3339418.

[126] Anne Roth, Sabine Ogrin, and Bernhard Schmitz. Assessing self-regulated learning in higher education: A systematic literature review of self-report instruments. *Educational Assessment, Evaluation and Accountability*, 28(3):225–250, 2016.

[127] Pooja Sankar. Our story, 2024. URL https://piazza.com/about/story.

[128] Gregory Schraw. Measuring self-regulation in computer-based learning environments. *Educational psychologist*, 45(4):258–266, 2010.

[129] Dale H Schunk and Barry Zimmerman. *Handbook of self-regulation of learning and performance*. Taylor & Francis, 2011.

[130] Clifford A. Shaffer and Ayaan M. Kazerouni. 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 AScience Education*, SIGCSE '21, page 907–913, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432356.

[131] Clifford A. Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L. Naps. Opendsa: Beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling '11, page 112–117, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450310529. doi: 10.1145/2094131.2094154. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/2094131.2094154.

[132] Kartik Shah, Priscilla Lee, Daphne Barretto, and Soohyun Nam Liao. A qualitative study on how students interact with quizzes and estimate confidence on their answers. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE '21, page 32–38, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382144. doi: 10.1145/3430665.3456377.

[133] Zalia Shams. *Automated Assessment of Student-written Tests Based on Defect-detection Capability*. PhD thesis, Virginia Tech, 2015.

[134] Melody Siadaty, Dragan Gasevic, and Marek Hatala. Trace-based micro-analytic measurement of self-regulated learning processes. *Journal of Learning Analytics*, 3(1): 183–214, 2016.

[135] Leonardo Silva, António José Mendes, Anabela Gomes, and Gabriel Fortes Cavalcanti de Macêdo. 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*, SIGCSE '21, page 647–653, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432363.

[136] Paramvir Singh, Ewan Tempero, Andrew Luxton-Reilly, and Shuxiang Zhang. What do they note? an exploratory investigation into the characteristics of cs students' notes. In *Proceedings of the 10th Computer Science Education Research Conference*, CSERC '21, page 57–67, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450385763. doi: 10.1145/3507923.3507934.

[137] Sharon Slade and Paul Prinsloo. Learning analytics: Ethical issues and dilemmas. *American Behavioral Scientist*, 57(10):1510–1529, 2013.

[138] Aaron J. Smith, Kristy Elizabeth Boyer, Jeffrey Forbes, Sarah Heckman, and Ketan Mayer-Patel. My digital hand: A tool for scaling up one-to-one peer teaching in support of computer science learning. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, page 549–554, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346986. doi: 10.1145/3017680.3017800. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3017680.3017800.

[139] Donggil Song, Hyeonmi Hong, and Eun Young Oh. Applying computational analysis of novice learners' computer programming patterns to reveal self-regulated learning, computational thinking, and learning performance. *Computers in Human Behavior*, 120:106746, 2021. doi: 10.1016/j.chb.2021.106746.

[140] Jaime Spacco, David Hovemeyer, William Pugh, Fawzi Emad, Jeffrey K. Hollingsworth, and Nelson Padua-Perez. Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. *SIGCSE Bull.*, 38(3):13–17, jun 2006. ISSN 0097-8418. doi: 10.1145/1140123.1140131. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/1140123.1140131.

[141] SPLICE. Splice: Standards, protocols, and learning infrastructure for computing education, 2024. URL https://cssplice.github.io/.

[142] Joel Spolsky. Painless software schedules. https://www.joelonsoftware.com/2000/03/29/painless-software-schedules/, March 2000.

[143] Kristin Stephens-Martinez. A study of the relationship between a cs1 student's gender and performance versus gauging understanding and study tactics. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, page 679–685, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432365.

[144] Ben Stephenson, Michelle Craig, Daniel Zingaro, Diane Horton, Danny Heap, and Elaine Huynh. Exam wrappers: Not a silver bullet. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, page 573–578, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346986. doi: 10.1145/3017680.3017701.

[145] Leigh Ann Sudol-DeLyser. Expression of abstraction: Self explanation in code production. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, page 272–277, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450329668. doi: 10.1145/2676723.2677222.

[146] Lauren C. Hensley Sungjun Won and Christopher A. Wolters. Brief research report: Sense of belonging and academic help-seeking as self-regulated learning. *The Journal of Experimental Education*, 89(1):112–124, 2021. doi: 10.1080/00220973.2019.1703095. URL https://doi.org/10.1080/00220973.2019.1703095.

[147] Lori Swinney and Amy Tichy. Discussion by design: Using discussion boards effectively. URL https://kb.ndsu.edu/page.php?id=131665.

[148] Michelle Taub and Roger Azevedo. How does prior knowledge influence eye fixations and sequences of cognitive and metacognitive srl processes during learning with an intelligent tutoring system? *International Journal of Artificial Intelligence in Education*, 29:1–28, 2019.

[149] DataShop Team. Datashop @cmu: a data analysis service for the learning science community, 2024. URL https://pslcdatashop.web.cmu.edu/.

[150] Adrian Thinnyun, Ryan Lenfant, Raymond Pettit, and John R. Hott. Gender and engagement in cs courses on piazza. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, page 438–444, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432395.

[151] Timothy Urness. Do students know what they think they know? assessing student confidence in a computer graphics course. *J. Comput. Sci. Coll.*, 31(5):140–146, May 2016. ISSN 1937-4771. doi: 10.5555/2904298.2904325.

[152] Joep Van Der Graaf, Lyn Lim, Yizhou Fan, Jonathan Kilgour, Johanna Moore, Maria Bannert, Dragan Gasevic, and Inge Molenaar. Do instrumentation tools capture self-regulated learning? In *LAK21: 11th international learning analytics and knowledge conference*, pages 438–448, 2021.

[153] Kevin Wang and Ramon Lawrence. Helpme: Student help seeking using office hours and email. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2024, page 1388–1394, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704239. doi: 10.1145/3626252.3630867. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3626252.3630867.

[154] Gerhard Weber and Peter Brusilovsky. Elm-art–an interactive and intelligent web-based electronic textbook. *International Journal of Artificial Intelligence in Education*, 26:72–81, 2016.

[155] C.E. Weinstein, D. Palmer, and A.C. Schulte. Learning and study strategies inventory (lassi). *Clearwater, FL: H H Publishing*, 1987.

[156] P. H. Winne and A. F. Hadwin. Studying as self-regulated engagement in learning. In *Metacognition in Educational Theory and Practice*, pages 277–304. 1998.

[157] Philip H Winne. A cognitive and metacognitive analysis of self-regulated learning. *Handbook of self-regulation of learning and performance*, pages 15–32, 2011.

[158] Philip H Winne. Learning analytics for self-regulated learning. *Handbook of learning analytics*, pages 241–249, 2017.

[159] Philip H Winne, Ryan SJd Baker, et al. The potentials of educational data mining for researching metacognition, motivation and self-regulated learning. *Journal of Educational Data Mining*, 5(1):1–8, 2013.

[160] Philip H Winne, Kenny Teng, Daniel Chang, Michael Pin-Chuan Lin, Zahia Marzouk, John C Nesbit, Alexandra Patzak, Mladen Raković, Donya Samadi, and Jovita Vytasek. nstudy: Software for learning analytics about processes for self-regulated learning. *Journal of Learning Analytics*, 6(2):95–106, 2019.

[161] xAPI. xapi, 2024. URL https://piazza.com/about/story.

[162] Jiayi Zhang, Taylor Cunningham, Rashmi Iyer, Ryan Baker, and Eric Fouh. Exploring the impact of voluntary practice and procrastination in an introductory programming course. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, SIGCSE 2022, page 356–361, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390705. doi: 10.1145/3478431.3499350.

[163] Barry J Zimmerman. Becoming a self-regulated learner: Which are the key subprocesses? *Contemporary educational psychology*, 11(4):307–313, 1986.

[164] Barry J Zimmerman. Investigating self-regulation and motivation: Historical background, methodological developments, and future prospects. *American educational research journal*, 45(1):166–183, 2008.

[165] Barry J Zimmerman and Adam R Moylan. Self-regulation: Where metacognition and motivation intersect. In *Handbook of metacognition in education*, pages 311–328. 2009. doi: 10.4324/9780203876428.ch16.

[166] Barry J Zimmerman and Manuel Martinez Pons. Development of a structured interview for assessing student use of self-regulated learning strategies. *American educational research journal*, 23(4):614–628, 1986. doi: 10.3102/00028312023004614.

[167] Zoom. Zoom: One platform for limitless human connection, 2024. URL https://zoom.us/.