frontiers | Frontiers in Education

TYPE Original Research
PUBLISHED 27 February 2025
DOI 10.3389/feduc.2025.1487344

OPEN ACCESS

EDITED BY
Octavian Dospinescu,
Alexandru Ioan Cuza University, Romania

REVIEWED BY
Xinghua Wang,
Qingdao University, China
Maila Pentucci,
University of Studies G. d'Annunzio Chieti and Pescara, Italy

*CORRESPONDENCE
Molly Domino
✉ m.domino@northeastern.edu

RECEIVED 28 August 2024
ACCEPTED 13 January 2025
PUBLISHED 27 February 2025

CITATION
Domino M and Shaffer CA (2025) Using a wider digital ecosystem to improve self-regulated learning.
Front. Educ. 10:1487344.
doi: 10.3389/feduc.2025.1487344

Using a wider digital ecosystem to improve self-regulated learning

Molly Domino[1]* and Clifford A. Shaffer[2]

[1]Department of Computer Science, Roux Institute of Northeastern University, Portland, ME, United States, [2]Department of Computer Science, Virginia Tech, Blacksburg, VA, United States

Introduction: Self-regulated learning skills are necessary for academic success. While not all students entering post-secondary education are proficient at many of these critical skills, they can be improved upon when practiced. However, self-regulation tends to be highly internal, making it difficult to measure. One form of measurement comes from using data traces collected from educational software. These allow researchers to make strong empirical inferences about a student's internal state. Automatically captured data traces also make it possible to provide automated interventions that help students practice and master self-regulated learning skills.

Methods/results: Using an experimental methodology we created a set of promising data traces that are grounded in theory to study self-regulated learning within a typical Computer Science course. Extra attention is given to studying the skill of help-seeking, which is both a key to success in CS and requires unobtrusive observation to properly measure.

Discussion: We also make the case for taking a broader perspective with our data collection efforts. The traces identified in this paper are not from one source, but the full ecosystem of software tools common to CS courses.

KEYWORDS

self-regulated learning, learning analytics, computing education, measurement models, data traces, unobtrusive

1 Introduction

Academic success requires not only knowing what to learn but also how to learn. Education research has found Self-Regulated Learning (SRL) to be a key predictor of academic success (Zimmerman and Martinez-Pons, 1990; Pintrich and De Groot, 1990; Lúftenegger et al., 2012), and it may be particularly critical in learning to program (Loksa and Ko, 2016; Flanigan et al., 2015; Sands and Yadav, 2020).

It is therefore no surprise that SRL has seen much study within the Computing Education Research (CER) community. Besides being a critical aspect of academic success, it is something that students can improve at when taught and practiced. However, even at the post-secondary level, studies show novice programmers' ability to self-regulate is "...inconsistent and shallow, but trainable through direct instruction" (Loksa et al., 2020). Additionally, there is emerging evidence that teaching SRL skills can help improve classroom equity, as interventions help struggling students in qualitatively and quantitatively different ways from students who are already succeeding within the class (Domino et al., 2024b; Ilves et al., 2018; Shaffer and Kazerouni, 2021; Denny et al., 2021).

Phrasing things in this way makes the problem appear straightforward: if self-regulation skills are both valuable and improve when taught, then the obvious next step is to create interventions to help students practice and develop these skills. At the most fundamental level, we need to iteratively evaluate students' proficiency level with a

skill, offer the chance to practice, and measure their growth. Yet successful SRL involves managing a large and diverse set of skills. We have previously studied which skills appear to be most important (Domino et al., 2024a) and which have been most heavily studied in CS (Domino et al., 2024b). But before we can create a student-oriented intervention, we need to know what skills that student most needs help with. Once we decide what to target, we need to know if our interventions have had the desired effect. These are both issues of measurement.

To know what a student needs, there has to be some way to assess and evaluate their current skill levels. Any population-based study requires some ability to assess how a student population is behaving both before and after the intervention. Thus, conducting a needs assessment and conducting an intervention both require some way to measure SRL. Measurement is challenging when studying SRL because it is highly cognitive. Observing SRL in CS is further complicated by the fact that students do a large fraction of their coursework outside of class.

While the measurement of self regulation has taken many forms (see Section 2), the collection and study of data traces has grown in popularity over the past two decades. According to Winne, a data trace is "a bit of ambient data that affords relatively strong inferences about one or more cognitive, affective, metacognitive, and motivational states and processes" (Winne, 2017). Actions like highlighting text, running a test suite for code, posting on a discussion board, or clicking on a hyperlink, could all be examples of a data trace. Ultimately, any interaction data that allows us to make a strong inference about a student's ability to self regulate could be considered a trace.

Such traces offer several advantages. They help address limitations of other forms of measurement by more accurately capturing a student's behavior *in-situ*. Data traces can offer educators a real-time look into the strengths and needs of a student body. This view can help educators identify at-risk students more effectively and help inform judgements about what targeted instruction or extra help a student may need. Additionally, data traces can be collected for each student regardless of class size, meaning that students in large classes could receive individualized attention or assessment they might not otherwise be able to get.

The field of Learning Analytics (LA) has made great strides, creating whole platforms that collect, analyze, and intervene on self regulation using this approach (Araka et al., 2020). One common tool used for trace-based study in LA is a dashboard where instructors and students can see visualizations of various metrics. When it comes to SRL, these metrics often relate to how a student engaged with a single piece of educational software like a Learning Management System (LMS). To help increase a student's awareness of their own behavior, this interaction data is then used to create personalized visualizations for each student. However, the analysis that such dashboards can provide to students is somewhat limited and cannot supply instructors or researchers with the raw material to combine data and build new interventions.

While there is no shortage of studies that focus on SRL (Winne et al., 2019; Davis et al., 2016; Alharbi et al., 2014), many reviews of the literature have been calling for a more theory-based approach. Several meta-reviews have recently critiqued a trend within LA-based research where traces are derived from existing data and then mapped onto theoretical models of SRL (Bodily and Verbert, 2017; Matcha et al., 2020; Galaige et al., 2022). This is described in further detail in Section 2, however it is clear that identifying a set of traces that prioritize theory in their creation is a critical next step.

With those critiques in mind, our research question is: **what high-quality data traces can we as a research community use to measure SRL?**

Grounding traces in existing SRL theory is a start to answering this research question, but we also seek to identify traces that were valid, reliable and equitable. Detailed definitions for these terms can be found in Section 4.

To answer this question, we adapt the experimental methodology used by Cristea et al. (2023). Rather than starting from existing data or technology and identifying where theory overlaps with what is measured, we start by identifying traces which are derived from an existing understanding of theory. Only once we have an ideal list of traces do we then evaluate how possible it is for us to measure them with existing technology.

Our work also focuses specifically on what skills are most important for Computer Science and the tools used in CS classrooms to study those skills. Thus, this paper first presents a set of data traces that come from an examination of SRL theory, rather than a specific technology. From there, we examine which of those traces are presently available to be measured.

In performing that evaluation, it became clear that such traces are often not only available using current LA data collection tools, but that there is an opportunity 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 any given self-regulation skill, we can use multiple data traces from a variety of relevant sources. Currently, research using data traces tends to gather those traces from a single source (typically an LMS) (Cristea et al., 2023; Alharbi et al., 2014; Arakawa et al., 2021; Arnold and Pistilli, 2012). The additional context that comes from collecting data from multiple sources offers the opportunity to provide a much more nuanced view into learning, allowing for better inferences to back to SRL skills. Thus, in addition to our work in identifying data traces, we also make the case that using multiple different tools from within a CS class's digital ecosystem is not only advantageous but highly possible using current technology.

To that end, this paper is structured as follows. Section 2 presents Winne and Hadwin's model of SRL, the trends in measuring SRL, and some working definitions for common terms that we use. Section 3 details the results of our previous studies wherein we identified a set of twelve SRL skills to prioritize for future research. In Section 4, we detail what we are looking for in an ideal trace and establish the types of data traces we are seeking to identify. Section 5 describes our approach to deriving high quality data traces for each skill, the results of which are summarized in Section 7. Section 6 evaluates a set of tools common to CS classrooms that could be valuable data sources for data traces and notes the ways in which each of those data sources might influence the traces they collect. Sections 8, 9 explores some areas that future researchers will need to be mindful of, the former discussing challenges that still need to be overcome and the latter exploring

some ethical considerations. We then conclude and discuss future work in Section 10.

2 Background

In this section we first provide our working definitions for SRL and related terms. We then discuss the three most common theoretical frameworks used to discuss SRL within the CS and LA research communities, and the broad trends in how researchers measure it.

2.1 Relevant vocabulary

Self regulation is closely connected to metacognition. The terms are sometimes used interchangeably and sometimes have their own distinct meanings, depending on the domain and the author's preference (Prather et al., 2020). For our purposes, we consider metacognition to be the knowledge generated from the process of thinking about thinking while self-regulation centers on the application of such knowledge. Under these definitions, only self-regulation can be observed and is therefore the sole focus of this study.

We consider a self-regulation **skill** to be an internal mastery of a particular aspect of self regulation. A **behavior** is an observable action or evidence that indicates the presence (or absence) of a particular SRL skill. For example, maintaining an awareness of one's remaining time to complete an assignment is a self-regulatory skill, while allocating time to study before an exam might be a behavioral indicator of strong time-management skill. This is important because any inference from a data point to an internal construct is ultimately an abstraction of what is happening. We use traces to make inferences on behaviors that allow us to make inferences on the presence of skills.

2.2 A brief overview of SRL theories

Panadero et al.'s (2016) meta-review identified six major models with different theoretical underpinnings. While discussions of Zimmerman's Cyclical Phase Model and Pintrich's more motivation-focused model are the two most commonly seen within the CER community, we will focus this section on the model we employ most within this study: Winne and Hadwin's COPES model.

The COPES model 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 each phase. The four phases are "task definition," "goal setting and planning," "enactment," and "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 five dimensions identified in this model are Conditions, Operations, Products, Evaluations, and Standards (COPES).

- Conditions are any variables that will influence the work.
- Operations are the ways a student processes information and behaves 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. The student 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. Within this work we use the COPES model to help us identify our proposed data traces.

2.3 Approaches to measuring SRL

The question of how to best measure SRL skills has proven difficult to answer. Panadero et al. (2016) 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 (Pintrich et al., 1991) or Learning and Study Strategies Inventory (LASSI) (Weinstein et al., 1987). As Prather et al. (2020) 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."

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 student behavior, 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. 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, in their Think-Aloud study (Loksa and Ko, 2016), Loksa and Ko used statements like "I'm going to initialize variables first" to indicate that a student was planning. With data traces, researchers need to ensure they have two inferences: first a strong inference from data to behavior, second a strong inference between that behavior to a skill. Thus researchers need to ensure that twice as many inferences are as strong as possible in order to collect meaningful data.

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 et al. (2016) 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. For researchers, the prose of the diary helps provide a view into how a student's self-regulatory processes develop 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 (Araka et al., 2020). Students get to immediately see trends in their behavior summed up. By tracking visits to those dashboards, researchers can keep tabs on how frequently a student reflects on their own processes.

2.4 An alternative approach to doing trace identification

As mentioned in Section 1, there has been no shortage of studies examining SRL through data traces over the last two decades. Systematic reviews by Bodily and Verbert (2017) and Matcha et al. (2020) both examine LA tools and their applications in student facing dashboards, though only Matcha et al. focused on the study of SRL using these tools. In 2020, Araka et al. (2020) published a detailed account of the SRL tools from 2008 to 2018. A more thorough examination of specific tools for studying SRL through data traces in a CS classroom can be found in Section 6

In their systematic reviews, however, these authors report a concerning trend where studies do not ground their data traces or findings to existing educational theory. Bodily and Verbert (2017) 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. Furthermore, Bodily and Verbert (2017) identified only 3 of those studies (3% of their full corpus) that "conducted a meaningful information selection process." It also appears that this lack of reporting has lead to a 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. (2020) was one of the key findings in their 2019 literature review and Galaige et al.'s (2022) systematic review took, if anything, an even darker view on the state of theory within LA design. 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 techno-centric design methods that focus on the availability of data with little attention to learning science theory" (Galaige et al., 2022).

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 generalize 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" (Gray and Bergner, 2022). Even in situations where available data is prioritized, such traces are going to be limited in their applicability. For example, Gašević et al. (2016) note a situation where two biology courses make use of embedded assessments within the Moodle LMS. 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. (2020) end up making the same recommendation, calling for traces to be derived from existing theory rather than availability.

Overall these meta-reviews indicate a need for a greater emphasis on theory. In this study, we seek to identify a set of data traces that come from an understanding of existing theoretical models of SRL and the needs of a post-secondary CS student. To do this, we drew upon previous research into what SRL skills are needed within the CS classroom (Domino et al., 2024a) and what SRL skills the research community has focused on (Domino et al., 2024b) to create a shortlist of SRL skills that we want to examine. From there, we began to identify how each of those skills could be measured through digital engagement. At time of writing, there are no existing recommendations on how to engage in such an approach. Therefore, we adapted the exploratory approach used by Cristea et al. (2023) in 2023, as that study also sought to derive SRL traces from existing literature (see Section 5 for further details).

3 Identifying skills

Before deriving data traces for SRL skills, we first must identify what aspects of self regulation we wish to measure. Table 1 outlines a list of SRL skills that we have identified in previous studies as most important to prioritize for success within CS. This table is the

TABLE 1 List of SRL skills and definitions.

Skillset	Skill	Definition
Planning	Task analysis	Forming an accurate conceptual model of the task at hand
	Scheduling	Intentionally allocating 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 their current emotional state and the potential impact those emotions could have on their 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
	Working with peers/ Group mates	Intentionally adapting the way one works to better facilitate the needs of a group
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	Knowledge building	Engaging in knowledge building in ways that are not for credit within a course

product of two previous research studies conducted by the authors. Domino et al. (2024a) asked post-secondary CS educators what SRL skills are most valuable for success by conducting a Delphi Process study. From this process, the panel of educators came to a consensus on 14 different self-regulation skills grouped into 5 distinct skillsets. Additionally, Domino et al. (2024b) conducted a systematic literature review seeking to identify SRL skills of interest to the CER community. In an attempt to help guide future research, Domino et al. also examined what relationship, if any, these skills had to academic success. Qualitative analysis of the 38 works studied within this review identified 11 skills.

Overall, 15 unique skills were identified across both studies. While there was overlap on most skills, there were some conflicting definitions and other discrepancies between the two. In resolving those changes, we refined the list to the 12 skills shown in Table 1. A detailed overview of how those discrepancies were resolved can be found in Domino (2024). The remainder of this section briefly presents this list of skills.

There are 5 broad skillsets that our skills are grouped into: Planning, Monitoring, Control, Reflection, and Phase-Independent. These adhere to Puustinen and Pulkkinen's (2001) meta-framework for common phases seen in self-regulation models.

The Planning skillset focuses on the ways in which a student prepares to work. Task Analysis encapsulates the ways in which a student forms an accurate conceptual model of an assignment. Decomposing focuses on how a student takes the abstract requirements of an assignment and breaks them down into more concrete sub-tasks. Scheduling concerns how that student allocates future time to work to complete the task.

The Monitoring and Control skillsets are both used while a student is executing a task. Monitoring concerns the ways in which a student observes themselves. These include the ways in which a student validates their work is correct (Monitoring Correctness), stays aware of their pace of work with respect to a deadline (Monitoring Progress) and the ways in which they stay aware of and work with their emotions (Emotional Regulation).

Control concerns the sort of actions that a student takes based on their self-monitoring. Our previous research identified two different aspects of help-seeking. First, Knowing How to Seek Help concerns the types of questions a student chooses to ask and what avenues they use to seek help. For instance, contrast the help received from Stack Overflow (which is available at all times but not necessarily correct) versus TA office hours (which are only available at specific times but likely more accurate and personal). Whereas TA office hours are limited to certain times and locations but can offer much more accurate and personalized explanations. Knowing How to Seek Help requires evaluating what resources are available and how to most effectively engage with those resources. Second, Knowing When to Seek Help concerns how a student balances working independently with seeking help. Marwan et al. identified that help seeking can be unproductive both if it is overly used or if it is avoided. Students show proficiency at Knowing When to Seek Help when they seek help after first making an attempt to solve the problem on their own for an appropriate period of time.

The skills in the Reflection skillset are used after a task is completed and a student is looking back on their work. Reflection Using Internal Standards refers to the ways in which a student evaluates their work compared to their expectations of themselves. Reflection Using External Standards focuses more on the ways in which a student evaluates their work compared to the standards established by the instructor.

One skill, Knowledge Building, could happen at any phase of the self-regulatory process and was therefore categorized in a unique skillset to reflect that: Phase-Independent. This skill encompasses all of the ways a student might explore the concepts discussed in a class that are not for credit. This includes behaviors like tinkering with code or asking tangential questions.

4 High-quality traces: concepts and characteristics

Given a list of SRL skills to explore more deeply, we next consider what qualities researchers should look for within the data traces that they use.

4.1 Three dimensions of quality

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

1. **Validity** considers how effectively the data gathered allows researchers to make an inference to an internal construct. Gray and Bergner (2022) ask: "Do questionnaire answers or facial expression actually measure *boredom*?" as an example of how one questions the validity of the inference between a metric and an internal state. Winne et al. (2019) note that traces should provide "an objective...account about how a learner operates on particular information at a point in time and in a relatively well-identified context." Ideally we would assess validity by validating a trace against some ground truth (a challenge we discuss in Section 8.1). For now, we largely assess whether 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** concerns how effectively a measure will capture the same thing in different contexts. Gray and Bergner (2022) define reliability as "repeatability or consistency of the instrument observations." To some degree, this comes down to how we select and standardize our metrics and thresholds. How many times should a student's code fail to compile before we classify them as being "stuck"? This sort of question is a matter of reliability within measurement.

   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. Portable data traces are desirable as they can be meaningfully used by different teams of researchers in different contexts (Cristea et al., 2023) 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 SRL skill among students, from those that are strong at SRL to those who struggle to master these skills. For example, the only students who likely will 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 usage data does not communicate much meaningful information about the students who are still learning to self-regulate effectively.

   Our systematic review (Domino et al., 2024b) found 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. This demonstrates the importance of considering equitability.

4.2 Data trace scope

Within this work, we seek to identify traces that provide instructors with raw material to observe skills without bias. Thus, we focus on micro-level, unobtrusive traces as they can provide the most detail and be used together to eventually make more macro-level inferences. By micro-level traces we mean those that can measure specific aspects of a single self-regulatory phase. These are contrasted with macro-level traces which can only identify a student is within a single SRL phase, but cannot draw further conclusions about what they are doing at any moment. For example a macro-level trace might be able to indicate a student is planning while a micro-level trace might indicate the specific nuances of the way a student breaks down a problem. The term 'unobtrusive' comes from Schraw's (2010) taxonomy of measurement approaches in education. In this paper, the authors separate measures that require a student's active attention (obtrusive measures) from those that do not (unobtrusive measures). We also focus on unobtrusive data traces as they capture authentic learning and cannot be biased toward a student's beliefs, like student self reports.

4.3 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. Additionally, 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. Further discussion on the advantages of using data traces together to create better inferences is discussed in more detail in Section 6.10.

5 Materials and methods

At the time of writing, there appears to be little guidance in the literature for selecting quality data traces. While there are frameworks for creating LA platforms (Siadaty et al., 2016; Matcha et al., 2020; Martinez-Maldonado et al., 2015), they focus on how work is analyzed. None of these offer much actionable guidance on how to select effective data traces to study.

Our approach loosely follows the experimental approach laid out by Cristea et al. (2023). 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 they also sought to ground their approach in existing SRL theory.

We started with the set of SRL skills outlined in Section 3 as most important to prioritize. Next, we used the COPES model to identify Operations and Products for each of those 12 skills. Then, we assessed the technologies common to the digital ecosystem of a CS classroom, and assessed the benefits and limitations of those data sources (see Section 6). Finally we mapped indicators to specific digital interactions, and arrived at a set of metrics that can provide a strong inference. The results of that mapping can be seen in Tables 2–4.

However, the approach we take in this paper is unique in a two major ways. First, we derive traces from all of the different digital education tools we can think of that are widely used in CS classes, whereas Cristea et al. (2023) gather data solely from an LMS. Second, Christea et al. start by identifying data traces that have been used in other studies, while our approach focuses on finding the best possible traces for each skill using the tools available. We chose to do this based on a recommendation from Matcha et al. (2020) who 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. reference, we found few works fully describe what traces they used in their studies and traces were rarely related back to SRL theory. This is consistent with Bodily and Verbert's (2017) meta-review of LA literature, discussed in Section 1. Therefore, we focus more on identifying the best possible indicators for each of our identified SRL skills.

6 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 managers
7. Specialized SRL support tools
8. Learning management systems

We note limitations and biases that researchers need to be aware of when using these traces. Following Section 4, each source will be evaluated in terms of the validity, reliability, and equitability of the traces that source can produce.

As a note, the goal of this section is to start a greater conversation on how we as a research community can start identifying these skills within their own classroom. This categorization is not intended to be exhaustive nor definitive. Instead, it is the author's perspective on what types of technologies exist currently and how those tools might help or hinder the measurement process.

6.1 Practice exercises

A wide variety of practice exercises are becoming prevalent in CS courses. These can range from small programming exercises (Parlante, 2017)[1] (Edwards and Murali, 2017; Hovemeyer and Spacco, 2013) to proficiency exercises that make students show the steps of an algorithm (Korhonen and Malmi, 2000; Shaffer et al., 2011) to basic batteries of multiple choice, fill-in-the-blank, and true/false questions.

Within CS classes, one common form of practice exercise asks 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 (Hovemeyer and Spacco, 2013) track keystroke-level data to see the process used by students as they write code. This provides a much more detailed source of information about a student's full process of solving a problem, possibly allowing for detection of misconceptions or misunderstandings.

Even in coding-focused classes, practice problems can extend beyond typing in code. Many CS classrooms make use of Parsons problems, which are 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 (Ericson et al., 2022a). But they can take more abstracted forms and represent concepts like loops or conditionals that are separate from syntax. They can also be used to let students practice a scaffolded form of writing proofs (Poulsen et al., 2022). As an example of a data trace, RuneStone (Ericson and Miller, 2020) logs of when a student starts the problem, each move of a line, and when they correctly complete the problem. Such data could then be related to a variety of academic outcomes. For example, in 2022 a study by Ericson et al. found a correlation between the number of moves taken while engaging with Parson's Problems on RuneStone and midterm scores (Ericson et al., 2022b).

Practice exercises can also take purely conceptual forms. OpenDSA (Shaffer et al., 2011), an e-textbook platform with practice exercise integration, makes use of interactive visualization questions, many similar to the "proficiency exercises" pioneered by the TRAKLA system (Korhonen and Malmi, 2000). 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. Programmed Instruction (Lockee et al., 2013) can be implemented in this way (Mohammed and Shaffer, 2024). With these problems, detailed logs of the state of the visualization are captured every time the user clicks on anything.

6.1.1 Evaluating practice exercises

There is a trade-off to consider when pulling data from practice exercises. Optional practice exercises offer a simple, yet highly valid

1 Parlante, N. (2017). *Codingbat code practice*.

TABLE 2 Operations, products, data traces, and sources for SRL skills related to planning.

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) Discussion board
		Record of student seeking clarification	Timestamp of when question was asked	
			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 3 Operations, products, data traces, and sources for SRL skills related to monitoring and control.

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 (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 toward 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		Timestamp of access to material	
			Duration of engagement with material	

(Continued)

TABLE 3 (Continued)

Skill	Operations	Products	Data traces	Data sources
		Records of strategies used before going to office hours	Qualitative attributes of that engagement	Page access event to content (e-textbook) or discussion board post or page access of course logistics
		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

Gray indicate they are behaviors with no corresponding data trace antecedent.

inference because students only 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. 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, optional tool traces gain 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 non-representative subset of students.

Requiring engagement with practice exercises has the opposite problem. Once engagement with practice problems is no longer optional, we gain a more equitable data set yet sacrifice that self-regulation inference. This trade-off is discussed more at length in Section 6.7, where all of the tools are optional.

Fortunately, required practice problems still offer effective data traces. Keystroke-level data [as collected by CloudCoder (Hovemeyer and Spacco, 2013)] 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 (Baker et al., 2004). Identified instances of 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 (specifically, portability) is a concern, 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 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.

TABLE 4 Operations, products, data traces, and sources for SRL skills related to reflection and knowledge building.

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)
	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	
Knowledge building	Practicing outside of the scope of the class	Engaged with practice items	Removed as it requires a self-report to differentiate between truly tangential questions and misunderstandings (and is therefore obtrusive)	N/A

Gray indicate they are behaviors with no corresponding data trace antecedent.

6.2 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 (Shaffer et al., 2011) and RuneStone (Ericson and Miller, 2020) 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 6.1) 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 (Weber and Brusilovsky, 2016) 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 can be captured.

While the research community has yet to explore how this type of data can be used to identify self-regulated learning behaviors, some initial identification into types of behaviors has already been done. Researchers have identified data patterns that indicate a student is engaging in types of "credit seeking behavior" (Fouh et al., 2014; Koh et al., 2018), wherein students minimize their engagement with a page of content while still recieiving credit for doing so.

6.2.1 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 only those who are particularly strong at SRL.

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 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 to 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 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.

6.3 IDEs

Integrated Development Environments (IDE) allow students to edit, compile, and run a program, and get feedback on results and various code quality metrics. IDEs have become nearly ubiquitous within CS classes that require programming. 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 (Kazerouni et al., 2017a) (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 6.1. 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 small code-writing exercise 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. In 2017, Kazerouni et al. (2017b) used DevEventTracker and found significant relationships between when a student worked and the correctness of that work. Additionally,in their conclusion, Kazerouni et al. (2017b) note that "DevEventTracker data is rich enough that this work barely touches the surface."

Johnson et al. (2004) 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.

6.3.1 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 toward 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.

6.4 Automated assessment tools

By their very nature, CS classes frequently require students to complete programming assignments. Programming is an inherently iterative activity. The act of programming 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 (Kazerouni et al., 2017b) as a formal process.

It is therefore no surprise that 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 impacts their grade in the course, meaning students can check their code without consequence as they work.

AATs automate at least some aspects 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 (Pärtel et al., 2013) 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 (Edwards and Perez-Quinones, 2008), ProgEdu (Chen et al., 2017), and Marmoset (Spacco et al., 2006) take this approach and commit all code to Git repositories. PruTutor (Das et al., 2016) and Edgar (Mekterovic et al., 2020) can 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 toward a fully correct answer. They can tell the story of how a student progressed from starting an assignment to finishing it. 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 environments (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 (Karnalim and Simon, 2021) and Web-CAT (Edwards and Perez-Quinones, 2008)] 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 (Messer et al., 2024). This can be valuable context to take into account when assessing work to see if a student effectively regulated their time. As described in Domino et al. (2024b), 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 proficient student who is busy but skilled at assessing the time needed to complete the work from a student who allocated time poorly and is rushing to finish. However, those two students would likely have noticeably different code quality trajectories in their submissions. Thus, traces could be especially valuable for better understanding submission patterns.

The study of data traces gathered by AATs is also, comparatively, more explored than some of the other technologies discussed in this section. Paiva et al.'s (2022) state of the art review into AATs included an examination of how different AATs were being used in the space of Learning Analytics, though not to study SRL specifically. Additionally, middleware tools to help process the data collected by AATs to better summarize a student's habits (Allevato et al., 2008). Yet, at time of writing, AATs have been used less frequently to study SRL skills. In 2018, Prather et al. (2018) explored the ways in which a student may struggle to use metacognition while engaging with the Athene. While helpful context into what students may struggle with, this study was qualitative in nature and used direct observation, not data traces, to make inferences about a student's self-regulatory ability. More recently, Arakawa et al. did just that used "fail[ing] to pass the same test case in three sequential commits & pushes to GitHub" (Arakawa et al., 2021) as one of three indicators that a student might be struggling to self-regulate. While Arakawa et al. did use AAT to identify at-risk students, the authors used these traces to then identify participants for a qualitative study. Thus, there appears to be an opportunity to build off of these studies and explore how SRL traces gathered by AATs relate back to student success.

6.4.1 Evaluating AATs

Reliability between class contexts remains an issue for AAT-based measures.

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 cannot expect that data collected in two different classes to lead to the same SRL inferences in the same way. For example, Arakawa et al. (2021) 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 (Irwin and Edwards, 2019). 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 writing a good test suite (Shams, 2015). Mutation testing (DeMillo et al., 1978), 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.

6.5 Discussion boards

Many classes make use of a private, text-based forum for course-specific conversations. Discussion boards are frequently used as a tool for students seeking help from their peers or course staff. Help-seeking skills are an important self-regulation skill (Domino et al., 2024a), but they are also difficult to observe (Domino et al., 2024b) directly. Especially among novices, a sense of belonging is a major influencer in how students seek help (Doebling and Kazerouni, 2021). Self-identification is often avoided by students who are concerned that they are asking "dumb" or personal questions (Ren et al., 2019). Thus, having an unobtrusive view into how students seek help might provide more accuracy. 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. Discussion board platforms often log 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 beginning to make use of data traces from discussion boards to examine help-seeking as well. Thinnyun et al. (2021) used metrics from Sankar (2024),[2] 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 (Swinney and Tichy).[3] 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.

6.5.1 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 asking a question. For example, if researchers counted all discussion posts as help-seeking events, they would potentially end up erroneously counting all textual answers to every question as help-seeking events. While developments in machine learning and natural language processing have made this easier (Kanjirathinkal et al., 2013), 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 toward 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 (Doebling and Kazerouni, 2021) and can be hesitant to appear foolish in front of peers (Ren et al., 2019), this can strongly influence which students choose to use this tool.

6.6 Office hour attendance managers

Office Hour queuing systems like MyDigitalHand (Smith et al., 2017) or HelpMe (Wang and Lawrence, 2024) can keep students organized and can be useful for students seeking assistance in office hours. Zoom (2024)[4] 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 office hour queue manager. 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.

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.

This may be another strong area for future research to explore as, at least within a Computing Education Research, there do not appear to be very many studies leveraging traces from systems like these to study SRL through data.

6.6.1 Evaluating office hour attendance managers

Noting when a student enters office hours is a trace with high validity. Students rarely enter office hours by accident, and typically come because they are seeking help. This makes the inference from

2 Sankar, P. (2024). *Our Story*.

3 Swinney, L., and Tichy, A. Discussion by design: Using discussion boards effectively.

4 Zoom (2024). *Zoom: One platform for limitless human connection*.

"office hour enqueue event" to "student is seeking help" very strong. This can help us know not only how frequently a student sought help over an assignment's timeline, but also 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 ways 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.

6.7 Specialized SRL support

This category encompasses a suite of tools that are specifically intended to help scaffold SRL. These systems are added onto other digital platforms to help students practice specific self-regulation skills. Van Der Graaf et al. (2021) 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 course, 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 an e-textbook system, a student could designate a specific block of time for work and be able to monitor their progress by looking at the timer. 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.

Learning Analytics researchers have introduced many such tools. Azevedo et al. (2009) 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 (Taub and Azevedo, 2019). Within CS, Edwards et al. (2015) also created a built-in planner by creating a digital version of Spolsky's painless schedule sheets tool (Spolsky, 2000), as part of an intervention on procrastination. Winne et al. (2019) 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 traces note 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 that area of self-regulation. Hsiao et al. (2017) 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.

6.7.1 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 typically 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. (2021) 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." 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 (but don't use it).

6.8 Learning management systems

LMSs like Canvas (LMS)[5] or Moodle (Moodle)[6] 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 (Consortium, 2024).[7] 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. 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 or integrator of tools, rather than a distinct tool in its own right.

5 LMS, C. *Canvas by instructure: World's #1 teaching and learning software*.

6 Moodle. Moodle.

7 Consortium, E. (2024). *Learning tools interoperability (LTI)*.

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

6.9 Summarizing findings

Table 5 summarizes our assessments of how each of these tools rates on Validity, Reliability, and Equity.

6.10 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 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 might 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. Using multiple sources together allows for an inference that none can make in isolation. For example, a log of when a student opened a page is 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.

Second, 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 lose this particular measure of 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.

7 Data traces for high-priority SRL skills

Our research question is: **what high-quality data traces can we as a research community use to measure SRL?** Tables 2–4 serve as the answers to that question by summarizing our list of SRL skills and how each skill could be measured through software already relatively common to CS classrooms. These tables should, ideally, act as a road map for future researchers wishing to do empirical studies of self-regulation within CS classes.

There were several cases where our current tools do not provide a way to measure Operations and Products 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 example, asking a tangential question is listed as an indicator of Knowledge Building in Table 4, but it requires that the student knows their question is tangential to the concepts discussed in class. A student who asks an only semi-related question could be indicating that they misunderstand the topic. In order to know why they asked a question, we would need to ask directly, necessitating an obtrusive self-report. Such measures are outside of the scope of this work. It is possible future sources of data will be able to capture such data, but at time of writing, we know of no way to record intent in this way.

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

7.1 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 illustrate 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 of Domino et al. (2024b), we note that

TABLE 5 Summary of validity, reliability, and equity assessment.

	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

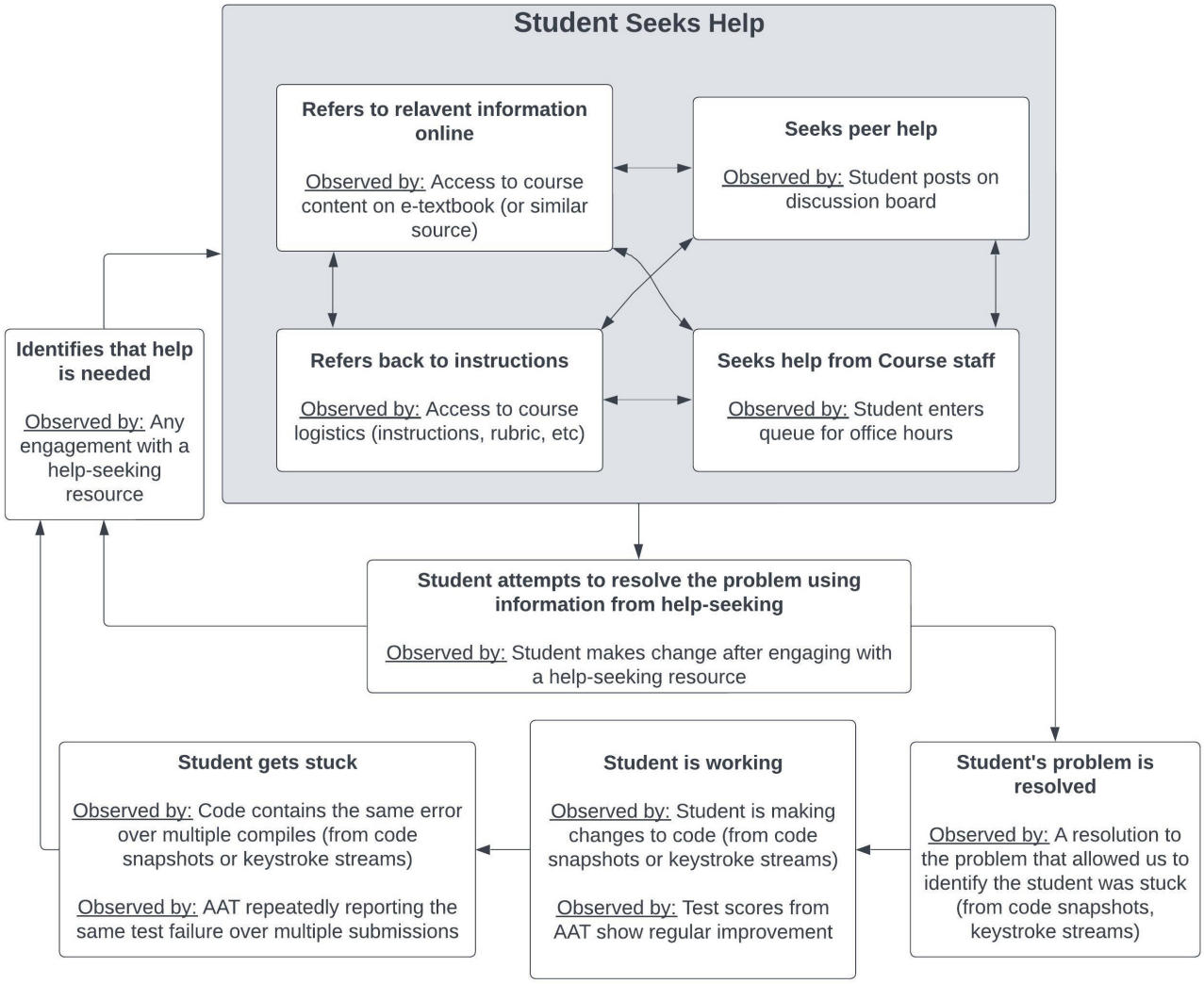

FIGURE 1
State diagram of help-seeking processes. See Section 7.1.

help-seeking remains a particularly difficult self-regulatory skill to examine, but also one that is highly important to success in CS (Domino et al., 2024a). 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 1 shows a state diagram created by the authors that details the phases a student goes through when seeking help when they encounter a problem while programming. The student eventually comes to a point where they are struggling to make progress toward a completed solution. They might then start seeking out resources to help them. Those resources could be entirely digital, like re-examining the instructions or going over a mis-remembered concept in a textbook. Doebling and Kazerouni (2021) observed 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 toward 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 place.

8 Additional challenges

In this section, we explore some challenges for future researchers seeking to extend this work.

8.1 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 largely internal. Within current literature, a ground truth is often established by checking metrics against an inventory of self-regulatory skills (Cristea et al., 2023), retention (Alharbi et al., 2014), or academic outcomes (Arnold and Pistilli, 2012). While these approaches are a good start toward validation, it is likely these data traces will require validation from a number of perspectives. These measures are approximations, and 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 a single source in isolation.

Similarly, since SRL is such a strong predictor of academic outcomes (Cheng et al., 2023), it may also be valuable to use 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.) to help triangulate what "success" really looks like. Self regulation, self efficacy, and motivation are so closely tied together, assessing for these related constructs is also important.

8.2 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 are ideally designed to protect identifying data, and it is a known issue that educational technologies do not integrate well (Brusilovsky et al., 2020). One major consequence 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) (Edwards and Perez-Quinones, 2008), OpenDSA (an e-textbook) (Shaffer et al., 2011), Piazza (a discussion board) (Sankar, 2024) (see text footnote 2), and Canvas (an LMS) (LMS) (see text footnote 5). 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 5,280 in Piazza. At time of writing, the 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, 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.

8.3 Privacy

Safeguarding personal information will need to be at the forefront of any future investigations. Depending on the system architecture, a given educational software system (like a coding exercise system) might or might not identify students using something like an email address that ought to be kept secure. Maintaining the privacy of personal identifiers requires that these systems block any access beyond anonymized identifiers. On the other hand, they also provide some level of coordination (typically through the LTI protocol) in order to allow the LMS to aggregate scores. Similar mechanisms could be used to connect data streams from multiple systems in a way that recognizes that a given individual is the same in each system, without releasing the actual identification of the individual. This would allow interventions that work to strengthen SRL skills without risking privacy violations.

8.4 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. 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 Sections 8.2, 8.3 could be ameliorated if student records from Piazza, OpenDSA, Web-CAT and Canvas were already using a standardized ID system for students, or were returning anonymized data traces directly to a coordinating central broker in a way that the LTI protocol allows for anonymized passing of scoring data. But beyond IDs, standardization requires that data traces take an expected format, making integration between systems or with central hubs (like an LMS or centralized data trace database) possible.

Fortunately, there have been significant gains in data standardization. Most notable at present is the LTI Standard (Consortium, 2024) (see text footnote 7), 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 Caliper (Kim and Ahn, 2016), which seeks a formalized structure for capturing learning activity data such as search activity, annotations, and forum activity. xAPI (xAPI, 2024)[8] 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 (SPLICE, 2024) 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 ProgSnap2 and PEML. ProgSnap2 (Price et al., 2020) captures snapshots of code, meant to trace the development of student solutions primarily in small programming exercises. PEML (Mishra and Edwards, 2023)

8 xAPI (2024). xapi.

provides a standard for defining small programming exercises with extensions to support 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 niche segments of the CER community, general standards like Caliper and xAPI are only slowly gaining traction. Thus, while progress is being made, the research community would benefit from a coordinated effort to adopt these standards.

9 Ethical concerns for data traces

We have presented a variety of data traces that can serve as a menu of opportunities for researchers to choose from including many ways that it is possible to track student data. However, our goal has never been to extract every possible metric out of our students in the process of gaining better insight. Fortunately, 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. We provide as thorough a study as we can about what trace data is available in order to guide future research efforts.

Within the remainder of this section we explore some of the ethical considerations for research within learning analytics. In doing so, we seek to ensure that future research not only has a menu of data sources and data traces with which to better study SRL, but also is well equipped to do such research ethically. Hakimi et al. (2021) noted four major categories of ethical issues for data traces in their 2021 systematic review: "privacy, informed consent, and data ownership; validity and integrity of data and algorithms; ethical decision making and the obligation to act; and governance and accountability." We next briefly summarize the nature of the first three of these issues as they are important considerations for future research, because they are important considerations for any researcher seeking to make use of the data sources we have explored in this paper. Governance and accountability, while important for ensuring ethical designs, is concerned more with regulating learning analytics at the institutional level, rather than at the scope of a single researcher or research team. Offering advice on what to do in light of these ethical concerns is beyond the scope of this paper.

Privacy, informed consent, and data ownership— Most educational software collects data on all students. Generally speaking, the burden of providing informed consent is on researchers who seek to make use of human interaction data. When such tools are not being used for research, traces are still being gathered, sometimes without informed consent. For example, Hakimi et al. (2021) discuss the idea of data dashboards to help support students, yet those same students (or parents) might not be aware of these dashboards or what data is collected to create them. This reflects a larger trend of "datafication" in our modern culture, where personal information is increasingly a normalized (and highly valuable) commodity. Who owns the data on student interaction? If data is owned by the student generating them, what power to control their information do they need? If data is owned by is the creator of the technology, what level of transparency do they owe their users regarding data collection and data security? This latter question is especially important considering some of these technologies are created by private vendors who have the power to prioritize monetization of data over ethics without regulation.

Validity and integrity of data and algorithms— Data traces are valuable tools to help researchers better understand the black box of a student's mind, but it is critical to remember that data traces are, at best, inferences. As Hakimi et al. (2021) note: "Big Data only tends to include information that is easily measurable and readily quantifiable... fixating on data that show what can be measured sometimes leads to a failure to remember that the information is, at best, a partial representation of what one wishes to know about." For example, studies often use the timestamp of when a student submits an assignment to infer how they manage their time (Ilves et al., 2018; Leinonen et al., 2021; Zhang et al., 2022). It is easy to see starting an assignment the night it is due as a student having procrastinated or otherwise starting too late. While this is certainly true for some of the population submitting work right before the due date, starting an assignment the night it is due might be an accurate allocation of time for students with high proficiency in the topic. The timestamp of when a student started an assignment may not provide enough context to differentiate between the student who procrastinated and the one who did not.

Ethical decision making and the obligation to act— Often in this paper, we have considered a best-case scenario where data are collected, cleaned, and clearly point toward a specific skill level. Yet it is also important to consider that data could be more nuanced than "fully clear" or "entirely obscure." Once data is collected that appears to point to a conclusion, how do researchers and educators decide to intervene given the "widespread acknowledgment that having access to data does not, per se, translate into complete or necessarily accurate information or knowledge" (Hakimi et al., 2021). When a student is at risk of dropping out of a class or receiving a failing grade, it seems clear that intervening is the best course of action, yet the same might not be true of a student who is at risk of their A grade lowering to an A-. Where is the line that propels an educator to intervene in a student's learning process?

10 Conclusion, limitations, and future work

In this paper, we have sought to identify high-quality SRL data traces that are grounded in theory. To accomplish this, we adapted an experimental methodology developed by Cristea et al. (2023). While we feel that the traces we have derived using that methodology are highly promising, an adaptation of an experimental methodology is, at its heart, still an experimental methodology. As we discussed in Section 8.1, validation of these proposed skills is the real next step. Furthermore, the other other hurdles laid out in Section 8 will slow that validation process down significantly. There is therefore still a long road before we will know for certain how effectively these data traces measure what we need them to. It is our hope, however that the research community is well-equipped to begin that process of validation.

In this work, we have also sought to make the case that utilizing data traces from a variety of sources is not only advantageous, but is already highly feasible given the educational software systems currently in widespread use in CS courses. To that end, we have detailed the types of tools commonly used within CS classes and provided an example of how one essential yet elusive SRL skill, help-seeking, could be studied under such a paradigm.

We believe research into SRL is at an exciting point in history. Although researchers have been exploring the area for decades, there is both a firm basis of things we do understand and many unknowns still left to explore. This means that SRL research currently offers space for creativity and new ideas in studying student learning.

It is our hope that future researchers can better utilize and begin to validate the data traces discussed in this paper to better explore and come to understand the process of how a student self regulates.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

MD: Conceptualization, Investigation, Methodology, Writing – original draft, Writing – review & editing. CS: Conceptualization, Investigation, Methodology, Project administration, Supervision, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Acknowledgments

Special thank yous to Conor Wallace and Dr. Alan Jamieson.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Alharbi, A., Henskens, F., and Hannaford, M. (2014). Personalised learning object system based on self-regulated learning theories. *Int. J. Eng. Pedagog.* 4, 24–35. doi: 10.3991/ijep.v4i3.3348

Allevato, A., Thornton, M., Edwards, S. H., and Pérez-Quiñones, M. A. (2008). "Mining data from an automated grading and testing system by adding rich reporting capabilities," in R. S. J. de Baker, T. Barnes, and J. E. Beck, eds. *Educational Data Mining 2008, The 1st International Conference on Educational Data Mining, Montreal, Québec, Canada, June 20–21, 2008* (Proceedings), 167–176.

Araka, E., Maina, E., Gitonga, R., and Oboko, R. (2020). Research trends in measurement and intervention tools for self-regulated learning for e-learning environments-systematic review (2008–2018). *Res. Pract. Technol. Enhan. Learn.* 15, 1–21. doi: 10.1186/s41039-020-00129-5

Arakawa, K., Hao, Q., Greer, T., Ding, L., Hundhausen, C. D., and Peterson, A. (2021). "In situ identification of student self-regulated learning struggles in programming assignments," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21* (New York, NY, USA: Association for Computing Machinery), 467–473. doi: 10.1145/3408877.3432357

Arnold, K. E., and Pistilli, M. D. (2012). "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* (New York, NY, USA: Association for Computing Machinery), 267–270. doi: 10.1145/2330601.2330666

Azevedo, R., Witherspoon, A., Chauncey, A., Burkett, C., and Fike, A. (2009). "Metatutor: a metacognitive tool for enhancing self-regulated learning," in *2009 AAAI Fall Symposium Series.*

Baker, R. S., Corbett, A. T., Koedinger, K. R., and Wagner, A. Z. (2004). "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* (New York, NY, USA: Association for Computing Machinery), 383–390. doi: 10.1145/985692.985741

Bodily, R., and Verbert, K. (2017). Review of research on student-facing learning analytics dashboards and educational recommender systems. *IEEE Trans. Learn. Technol.* 10, 405–418. doi: 10.1109/TLT.2017.2740172

Brusilovsky, P., Koedinger, K., Joyner, D. A., and Price, T. W. (2020). "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* (New York, NY, USA: Association for Computing Machinery), 211–213. doi: 10.1145/3386527.3405936

Chen, H.-M., Chen, W.-H., Hsueh, N.-L., Lee, C.-C., and Li, C.-H. (2017). "Progedu - an automatic assessment platform for programming courses," in *2017 International Conference on Applied System Innovation (ICASI)*, 173–176. doi: 10.1109/ICASI.2017.7988376

Cheng, Z., Zhang, Z., Xu, Q., Maeda, Y., and Gu, P. (2023). A meta-analysis addressing the relationship between self-regulated learning strategies and academic performance in online higher education. *J. Comput. High. Educ.* 42, 1–30. doi: 10.1007/s12528-023-09390-1

Cristea, T., Snijders, C., Matzat, U., and Kleingeld, A. (2023). Unobtrusive measurement of self-regulated learning: a clickstream-based multi-dimensional scale. *Educ. Inf. Technol.* 29, 13465–13494. doi: 10.1007/s10639-023-12372-6

Das, R., Ahmed, U. Z., Karkare, A., and Gulwani, S. (2016). Prutor: a system for tutoring CS1 and collecting student programs for analysis. *CoRR, abs/1608.03828.*

Davis, D., Chen, G., Jivet, I., Hauff, C., and Houben, G.-J. (2016). "Encouraging metacognition self-regulation in moocs through increased learner feedback," in *LAL@LAK*, 17–22.

DeMillo, R. A., Lipton, R. J., and Sayward, F. G. (1978). Hints on test data selection: help for the practicing programmer. *Computer* 11, 34–41. doi: 10.1109/C-M.1978.218136

Denny, P., Whalley, J., and Leinonen, J. (2021). "Promoting early engagement with programming assignments using scheduled automated feedback," in *Proceedings of the 23rd Australasian Computing Education Conference, ACE '21* (New York, NY, USA: Association for Computing Machinery), 88–95. doi: 10.1145/3441636.3442309

Doebling, A., and Kazerouni, A. M. (2021). "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: Association for Computing Machinery). doi: 10.1145/3488042.3488052

Domino, M. (2024). *Self-Regulated Learning Skills Research in Computer Science: The State of the Field*. Phd thesis, Virginia Tech, Blacksburg, VA.

Domino, M., Edmison, B., Edwards, S., Mansur, R., Thompson, A., and Shaffer, C. (2024a). Identifying critical self-regulated learning skills: a delphi process study. *Comput. Sci. Educ.* 15:2382631. doi: 10.1080/08993408.2024.2382631

Domino, M., Thompson, A., Hicks, A., Jamieson, A., Parajuli, K., Edmison, B., et al. (2024b). How can we know when we see it? A systematic review of cognitive control skills and behaviors. *ACM Trans. Comput. Educ.* (submitted for review).

Edwards, S. H., Martin, J., and Shaffer, C. A. (2015). "Examining classroom interventions to reduce procrastination," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '15* (New York, NY, USA: Association for Computing Machinery), 254–259. doi: 10.1145/2729094.2742632

Edwards, S. H., and Murali, K. P. (2017). "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* (New York, NY, USA: Association for Computing Machinery), 188–193. doi: 10.1145/3059009.3059055

Edwards, S. H., and Perez-Quinones, M. A. (2008). Web-cat: automatically grading programming assignments. *SIGCSE Bull.* 40:328. doi: 10.1145/1597849.1384371

Ericson, B. J., Denny, P., Prather, J., Duran, R., Hellas, A., Leinonen, J., et al. (2022a). "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* (New York, NY, USA: Association for Computing Machinery), 191–234. doi: 10.1145/3571785.3574127

Ericson, B. J., Maeda, H., and Dhillon, P. S. (2022b). "Detecting struggling students from interactive ebook data: A case study using csawesome," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1, SIGCSE 2022* (New York, NY, USA: Association for Computing Machinery), 418–424. doi: 10.1145/3478431.3499354

Ericson, B. J., and Miller, B. N. (2020). "Runestone: a platform for free, on-line, and interactive ebooks," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20* (New York, NY, USA: Association for Computing Machinery), 1012–1018. doi: 10.1145/3328778.3366950

Flanigan, A. E., Peteranetz, M. S., Shell, D. F., and Soh, L.-K. (2015). "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* (New York, NY, USA: Association for Computing Machinery), 161–168. doi: 10.1145/2787622.2787722

Fouh, E., Breakiron, D. A., Hamouda, S., Farghally, M. F., and Shaffer, C. A. (2014). Exploring students learning behavior with an interactive etextbook in computer science courses. *Comput. Hum. Behav.* 41, 478–485. doi: 10.1016/j.chb.2014.09.061

Galaige, J., Steele, G. T., Binnewies, S., and Wang, K. (2022). A framework for designing student-facing learning analytics to support self-regulated learning. *IEEE Trans. Learn. Technol.* 15, 376–391. doi: 10.1109/TLT.2022.3176968

Gašević, D., Dawson, S., Rogers, T., and Gasevic, D. (2016). Learning analytics should not promote one size fits all: the effects of instructional conditions in predicting academic success. *Internet High. Educ.* 28, 68–84. doi: 10.1016/j.iheduc.2015.10.002

Gray, G., and Bergner, Y. (2022). "A practitioner's guide to measurement in learning analytics: decisions, opportunities, and challenges," in *Handbook of Learning Analytics*, 20–28. doi: 10.18608/hla22.002

Hakimi, L., Eynon, R., and Murphy, V. A. (2021). The ethics of using digital trace data in education: a thematic review of the research landscape. *Rev. Educ. Res.* 91, 671–717. doi: 10.3102/00346543211020116

Hovemeyer, D., and Spacco, J. (2013). Cloudcoder: a web-based programming exercise system. *J. Comput. Sci. Coll.* 28:30.

Hsiao, I.-H., Huang, P.-K., and Murphy, H. (2017). "Uncovering reviewing and reflecting behaviors from paper-based formal assessment," in *Proceedings of the Seventh International Learning Analytics and Knowledge Conference, LAK '17* (New York, NY, USA: Association for Computing Machinery), 319–328. doi: 10.1145/3027385.3027415

Ilves, K., Leinonen, J., and Hellas, A. (2018). "Supporting self-regulated learning with visualizations in online learning environments," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18* (New York, NY, USA: Association for Computing Machinery), 257–262. doi: 10.1145/3159450.3159509

Irwin, M. S., and Edwards, S. H. (2019). "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* (New York, NY, USA: Association for Computing Machinery), 208–214. doi: 10.1145/3300115.3309517

Johnson, P., Kou, H., Agustin, J., Zhang, Q., Kagawa, A., and Yamashita, T. (2004). "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*, 136–144. doi: 10.1109/ISESE.2004.1334901

Kanjirathinkal, R. C., Gangadharaiah, R., Visweswariah, K., and Raghu, D. (2013). "Semi-supervised answer extraction from discussion forums," in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 1–9.

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

Kazerouni, A. M., Edwards, S. H., Hall, T. S., and Shaffer, C. A. (2017a). "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* (New York, NY, USA: Association for Computing Machinery), 104–109. doi: 10.1145/3059009.3059050

Kazerouni, A. M., Edwards, S. H., and Shaffer, C. A. (2017b). "Quantifying incremental development practices and their relationship to procrastination," in *Proceedings of the 2017 ACM Conference on International Computing Education Research, ICER '17* (New York, NY, USA: Association for Computing Machinery), 191–199. doi: 10.1145/3105726.3106180

Kim, Y. H., and Ahn, J.-H. (2016). A study on the application of big data to the korean college education system. *Procedia Comput. Sci.* 91, 855–861. doi: 10.1016/j.procs.2016.07.096

Koh, K. H., Fouh, E., Farghally, M. F., Shahin, H., and Shaffer, C. A. (2018). Experience: learner analytics data quality for an etextbook system. *J. Data Inf. Qual.* 9, 1–10. doi: 10.1145/3148240

Korhonen, A., and Malmi, L. (2000). "Algorithm simulation with automatic assessment," in *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSEconference on Innovation and Technology in Computer Science Education, ITiCSE '00* (New York, NY, USA: Association for Computing Machinery), 160–163. doi: 10.1145/343048.343157

Leinonen, J., Castro, F. E. V., and Hellas, A. (2021). "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* (New York, NY, USA: Association for Computing Machinery), 373–379. doi: 10.1145/3430665.3456383

Lockee, B., Moore, D. M., and Burton, J. (2013). "Foundations of programmed instruction," in *Handbook of Research on Educational Communications and Technology* (Routledge), 546–570.

Loksa, D., and Ko, A. J. (2016). "The role of self-regulation in programming problem solving process and success," in *Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16* (New York, NY, USA: Association for Computing Machinery), 83–91. doi: 10.1145/2960310.2960334

Loksa, D., Xie, B., Kwik, H., and Ko, A. J. (2020). *Investigating Novices' In Situ Reflections On Their Programming Process 149–155*. New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3328778.3366846

Lüftenegger, M., Schober, B., Schoot, R., Wagner, P., Finsterwald, M., and Spiel, C. (2012). Lifelong learning as a goal - do autonomy and self-regulation in school result in well prepared pupils? *Learn. Instr.* 22, 27–36. doi: 10.1016/j.learninstruc.2011.06.001

Martinez-Maldonado, R., Pardo, A., Mirriahi, N., Yacef, K., Kay, J., and Clayphan, A. (2015). Latux: an iterative workflow for designing, validating, and deploying learning analytics visualizations. *J. Learn. Anal.* 2, 9–39. doi: 10.18608/jla.2015.23.3

Matcha, W., Uzir, N. A., Gašević, D., and Pardo, A. (2020). A systematic review of empirical studies on learning analytics dashboards: a self-regulated learning perspective. *IEEE Trans. Learn. Technol.* 13, 226–245. doi: 10.1109/TLT.2019.2916802

Mekterović, I., Brkić, L., Milašinović, B., and Baranović, M. (2020). Building a comprehensive automated programming assessment system. *IEEE Access* 8, 81154–81172. doi: 10.1109/ACCESS.2020.2990980

Messer, M., Brown, N. C. C., Kölling, M., and Shi, M. (2024). Automated grading and feedback tools for programming education: a systematic review. *ACM Trans. Comput. Educ.* 24, 1–43. doi: 10.1145/3636515

Mishra, D. S., and Edwards, S. H. (2023). "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* (New York, NY, USA: Association for Computing Machinery), 395–401. doi: 10.1145/3545945.3569734

Mohammed, M., and Shaffer, C. A. (2024). "Teaching formal languages through programmed instruction," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*, 867–873. doi: 10.1145/3626252.3630940

Paiva, J. C., Leal, J. P., and Figueira, A. (2022). Automated assessment in computer science education: a state-of-the-art review. *ACM Trans. Comput. Educ.* 22, 1–40. doi: 10.1145/3513140

Panadero, E., Klug, J., and Järvelä, S. (2016). Third wave of measurement in the self-regulated learning field: when measurement and intervention come hand in hand. *Scandinavian J. Educ. Res.* 60, 723–735. doi: 10.1080/00313831.2015.1066436

Pärtel, M., Luukkainen, M., Vihavainen, A., and Vikberg, T. (2013). Test my code. *Int. J. Technol. Enhanc. Learn.* 5, 271–283. doi: 10.1504/IJTEL.2013.059495

Pintrich, P. R., et al. (1991). *A manual for the use of the motivated strategies for learning questionnaire (mslq)*. Technical report. Institute of Education Sciences. doi: 10.1037/t09161-000. Available at: https://eric.ed.gov/?id=ED338122

Pintrich, P. R., and De Groot, E. V. (1990). Motivational and self-regulated learning components of classroom academic performance. *J. Educ. Psychol.* 82:33. doi: 10.1037//0022-0663.82.1.33

Poulsen, S., Viswanathan, M., Herman, G. L., and West, M. (2022). "Proof blocks: Autogradable scaffolding activities for learning to write proofs," in *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1, ITiCSE '22* (New York, NY, USA: Association for Computing Machinery), 428–434. doi: 10.1145/3502718.3524774

Prather, J., Becker, B. A., Craig, M., Denny, P., Loksa, D., and Margulieux, L. (2020). "What do we think we think we are doing? metacognition and self-regulation in programming," in *Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER '20* (New York, NY, USA: Association for Computing Machinery), 2–13. doi: 10.1145/3372782.3406263

Prather, J., Pettit, R., McMurry, K., Peters, A., Homer, J., and Cohen, M. (2018). "Metacognitive difficulties faced by novice programmers in automated assessment tools," in *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER '18* (New York, NY, USA: Association for Computing Machinery), 41–50. doi: 10.1145/3230977.3230981

Price, T. W., Hovemeyer, D., Rivers, K., Gao, G., Bart, A. C., Kazerouni, A. M., et al. (2020). "Progsnap2: A flexible format for programming process data," in *ITiCSE '20* (New York, NY, USA: Association for Computing Machinery), 356–362. doi: 10.1145/3341525.3387373

Puustinen, M., and Pulkkinen, L. (2001). Models of self-regulated learning: a review. *Scandinavian J. Educ. Res.* 45, 269–286. doi: 10.1080/00313830120074206

Ren, Y., Krishnamurthi, S., and Fisler, K. (2019). "What help do students seek in ta office hours?" in *Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER '19* (New York, NY, USA: Association for Computing Machinery), 41–49. doi: 10.1145/3291279.3339418

Sands, P., and Yadav, A. (2020). *Self-Regulation For High School Learners In A Mooc Computer Science Course 845–851*. Association for Computing Machinery, New York, NY, USA. doi: 10.1145/3328778.3366818

Schraw, G. (2010). Measuring self-regulation in computer-based learning environments. *Educ. Psychol.* 45, 258–266. doi: 10.1080/00461520.2010.515936

Shaffer, C. A., Karavirta, V., Korhonen, A., and Naps, T. L. (2011). "Opendsa: Beginning a community active-ebook project," in *Proceedings of the 11th Koli Calling International Conference on Computing Education Research, Koli Calling '11* (New York, NY, USA: Association for Computing Machinery), 112–117. doi: 10.1145/2094131.2094154

Shaffer, C. A., and Kazerouni, A. M. (2021). "The impact of programming project milestones on procrastination, project outcomes, and course outcomes: a quasi-experimental study in a third-year data structures course," in *Proceedings of the 52nd ACM Technical Symposium on Computer AScience Education, SIGCSE '21* (New York, NY, USA: Association for Computing Machinery), 907–913. doi: 10.1145/3408877.3432356

Shams, Z. (2015). *Automated Assessment of Student-written Tests Based on Defect-detection Capability*. PhD thesis, Virginia Tech.

Siadaty, M., Gasevic, D., and Hatala, M. (2016). Trace-based micro-analytic measurement of self-regulated learning processes. *J. Learn. Analy.* 3, 183–214. doi: 10.18608/jla.2016.31.11

Smith, A. J., Boyer, K. E., Forbes, J., Heckman, S., and Mayer-Patel, K. (2017). "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* (New York, NY, USA: Association for Computing Machinery), 549–554. doi: 10.1145/3017680.3017800

Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J. K., and Padua-Perez, N. (2006). Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. *SIGCSE Bull.* 38, 13–17. doi: 10.1145/1140123.1140131

SPLICE (2024). *Splice: Standards, protocols, and learning infrastructure for computing education*. Available at: https://cssplice.github.io

Spolsky, J. (2000). *Painless software schedules*. Available at: https://www.joelonsoftware.com/2000/03/29/painless-software-schedules/ (accessed June, 2024).

Taub, M., and Azevedo, R. (2019). How does prior knowledge influence eye fixations and sequences of cognitive and metacognitive srl processes during learning with an intelligent tutoring system? *Int. J. Artif. Intell. Educ.* 29, 1–28. doi: 10.1007/s40593-018-0165-4

Thinnyun, A., Lenfant, R., Pettit, R., and Hott, J. R. (2021). "Gender and engagement in cs courses on piazza," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21* (New York, NY, USA: Association for Computing Machinery), 438–444. doi: 10.1145/3408877.3432395

Van Der Graaf, J., Lim, L., Fan, Y., Kilgour, J., Moore, J., Bannert, M., et al. (2021). "Do instrumentation tools capture self-regulated learning?" in *LAK21: 11th International Learning Analytics and Knowledge Conference*, 438–448. doi: 10.1145/3448139.3448181

Wang, K., and Lawrence, R. (2024). "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* (New York, NY, USA: Association for Computing Machinery), 1388–1394. doi: 10.1145/3626252.3630867

Weber, G., and Brusilovsky, P. (2016). Elm-art-an interactive and intelligent web-based electronic textbook. *Int. J. Artif. Intell. Educ.* 26, 72–81. doi: 10.1007/s40593-015-0066-8

Weinstein, C., Palmer, D., and Schulte, A. (1987). *Learning and Study Strategies Inventory (LASSI)*. Clearwater, FL: H and H Publishing.

Winne, P. H. (2017). Learning analytics for self-regulated learning. *Handb. Learn. Anal.* 754, 241–249. doi: 10.18608/hla17.021

Winne, P. H., Teng, K., Chang, D., Lin, M. P.-C., Marzouk, Z., Nesbit, J. C., et al. (2019). nstudy: Software for learning analytics about processes for self-regulated learning. *J. Learn. Anal.* 6, 95–106. doi: 10.18608/jla.2019.62.7

Zhang, J., Cunningham, T., Iyer, R., Baker, R., and Fouh, E. (2022). "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* (New York, NY, USA: Association for Computing Machinery), 356–361. doi: 10.1145/3478431.3499350

Zimmerman, B. J., and Martinez-Pons, M. (1990). Student differences in self-regulated learning: Relating grade, sex, and giftedness to self-efficacy and strategy use. *J. Educ. Psychol.* 82, 51. doi: 10.1037/0022-0663.82.1.51