Journal for STEM Education Research
https://doi.org/10.1007/s41979-025-00155-2

RESEARCH ARTICLE

Social Computational Literacy in Practice: A Framework Describing STEM Researchers' Communication

Christian Cammarota[1,2] · Michael Foster[1,3] · Mike Verostek[2,4] · Kayleigh Patterson[2] · Mikayla MacIntyre[3] · Kimberly Dorsey[5] · Andrea Camacho-Betancourt[6] · Tony E. Wong[3] · Benjamin Zwickl[2,7]

Received: 22 November 2024 / Revised: 19 June 2025 / Accepted: 19 June 2025

Abstract

Computational thinking is crucial for STEM researchers and practitioners, as it involves more than just developing skills—it is a way of thinking that enables effective problem-solving. STEM disciplines approach different problems and as such employ computational thinking uniquely, so students cannot rely solely on computer science to develop computational thinking. Less attention has been given to social aspects of computation, such as collaborating and communicating with and about computation even though social aspects are essential to problem solving. We utilized computational literacy as an alternative framework that explicitly includes social elements as a primary pillar. We conducted 15 interviews with STEM researchers to identify and organize the social aspects that play a role in their research. We organized goals by motivation (persuasion and productivity) and representation (visual and non-visual) to contextualize the use of communication in computation. We found that researchers use computation to explain research results, navigate decision making, establish rigor, ensure reproducibility, facilitate lab stability, and promote research efficiency. We used Activity Theory to describe the tools, norms, and communities associated with these goals to offer a more detailed framework for the social pillar of computational literacy within the context of science and engineering. Examples from each discipline within STEM are described. This social computational literacy framework can act as a guide for STEM educators and practitioners alike to use and teach social aspects of computation.

Keywords Computational literacy · Computational thinking · STEM communication · Activity theory · Social practices · Computational education

Extended author information available on the last page of the article

Published online: 22 July 2025

Introduction

The U.S. Next Gen Science Standards highlight computation as a key aspect in all fields of Science, Technology, Engineering, and Mathematics (STEM) (NGSS Lead States, 2013). In this work, we describe computation as more than just coding in a traditional sense but also the use of a variety of other information processing tools (e.g., Microsoft Excel and Graphing Calculators) (diSessa, 2001; Li et al., 2020a).

Within STEM, computation is particularly important for solving a variety of problems and is becoming a focus for instructors. However, the adoption of computation within STEM education varies by discipline. Computation is considered the third pillar of physics in addition to theory and experiment (AAPT Skuse, 2019; UCTF, 2016), while biology is only more recently incorporating computation into their curricula (as evidenced in Bialek & Botstein, 2004; A. Juavinett, 2020; A. L. Juavinett, 2022; Pevzner & Shamir, 2009). Calls for research and education reform highlight the necessity of computation in the classroom (e.g., Lockwood & Mørken, 2021). In their call, Lockwood and Mørken (2021) pose multiple research questions, such as the identification of topics suitable for programming integration, effective ways to support student engagement, and the coordination of computing between scientific departments. Concordant with these research questions are recent editorials from Li et al. (2020a), (2020b). Li et al. posit that the skills associated with computation, namely those included in computational thinking (CT), are useful to all STEM disciplines and should not be siloed within computer science.

While STEM fields are often valued for specialized knowledge and problem-solving abilities, communication and collaboration are also paramount. Multiple government agencies list communication as a key skill for scientists (Aizenman et al., 2024; American Association for the Advancement of Science, 1994; NGSS Lead States, 2013). Aside from these agencies, employability can depend on communication. The Accreditation Board for Engineering and Technology (ABET) requires accredited schools to teach effective communication to multiple audiences and employ faculty competent in communication (ABET, 2024). Furthermore, employers have focused on the need for communication skills in their employees (Carnevale et al., 2011; Finley, 2023).

Though often relegated to the background, some researchers have elevated communication and visualization through computation in their work. For instance, diSessa (2001) called out the collaborative nature of computation as a *social* pillar supporting computational literacy. Research is needed to investigate how and why communicative elements of computation are employed in practice. Furthermore, research is needed to supplement syntactical computational education with descriptions of coding in practice. With diSessa's framework as a guide, we set out to answer two questions: RQ1) What goals of STEM researchers are supported by social computational literacy?, and RQ2) How do STEM researchers implement social computational literacy to accomplish those goals? This work will identify the goals and implementations of social computational literacy in research and can help instructors orient learning objectives in their courses.

Literature Review

One lens into solving problems computationally and the associated thought processes is through CT. In this literature review, we start by describing CT; how CT has been applied to STEM; and how communication has been integrated into some CT frameworks. Additionally, we describe computational literacy (CL), a complementary framework that highlights the social aspects of computation. Finally, we give a brief description of Activity Theory (AT), a framework used in our data analysis and interpretation.

Computational Thinking in STEM

Wing (2006) described CT as "a fundamental skill for everyone, not just for computer scientists" (p. 33). Wing described CT as abstractly/efficiently solving problems and the application of recursion/heuristics to test and push the limits of computational solutions. Wing's focus remained on cognition as she went on to refine this definition to focus on problem solving by creating solutions that can be performed algorithmically (Lodi, 2020; Wing, 2008). Since Wing, other groups have provided definitions of CT with a significant focus on cognition as opposed to computational practices (e.g., Grover & Pea, 2013; Kalelioğlu et al., 2016). Computational practices are distinct from thinking in that practices define specific actions that are taken to write and execute code, such as debugging, 'tinkering,' and modularizing solutions.

There are efforts to emphasize the computational practices associated with CT. For instance, the International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA) operationalized a definition of CT that includes data collection, analysis, and visual creation in addition to the Wing-derived definitions associated with thinking (ISTE & CSTA, 2011). Similarly, Weintrop et al. (2016) compiled a list of CT practices into a taxonomy and briefly described how each of those practices exist within STEM. These practices were identified through a systematic literature review of CT in STEM, interviews with STEM practitioners, and coding of classroom activities.

Many studies have tried to identify unique formulations of CT for each discipline in STEM (Wang et al., 2022). In their review, Wang et al. analyzed 55 articles and reported on the operational definitions of CT and found that despite some variations, most formulations of CT focus on problem solving through abstraction, decomposition, and generalization, and algorithmic thinking. Furthermore, all STEM disciplines include practices, such as data analysis, modeling, and systems thinking (Beheshti et al., 2017). Though these results suggest that there is a common set of high-level computational practices for STEM, practices such as data analysis can have major variations when comparing disciplines (i.e., bioinformatics vs. gravitational wave observatories). In their study on CT in biology-engineering course, DeJarnette et al., (2022) found differences in how students employed CT by lab type. In a biology lab within the course, students were able to effectively engage in CT

practices such as using modeling and investigating systems without writing code, while the engineering lab elicited use of coding practices such as troubleshooting and creating abstractions.

Computational Thinking and Communication

The presence of some degree of communication within CT is clear, though indirect. In Wing's (2006) viewpoint article, she mentions communicative elements such as code aesthetics and anticipating multiple users. Aesthetic or elegant code provides readers with a reduced mental load as compared to disorganized or confusing code. Likewise, the act of anticipating a multi-user project implies an acknowledgment that computation is collaborative. Aside from Wing, in their literature review Juškevičienė and Dagienė (2018) relate CT with digital competence. Digital competence includes aspects such as communication, collaboration, ethics, teaching, professional engagement, and assessment. Juškevičienė and Dagienė find these communicative elements of computation present in various aspects of CT (e.g., data analysis, representation, and abstraction). These examples along with the general definitions put forth by Wing show communication is acknowledged but not given the level of attention as the cognitive aspects (decomposition, algorithmic thinking, etc.) (Juškevičienė & Dagienė, 2018; Wing, 2006).

Similarly, within a STEM specific context, communication is not directly described but rather indirectly included within CT practices. In Weintrop et al.'s (2016) taxonomy, several practices were social in nature. For example, data visualization, designing computational models (communicating ideas or principles related to a problem), and communicating information about a system (summarizing complex data to create tractable explanations) all involve communication. Other CT practices obliquely imply the communicative and collaborative nature of computation, such as data collection, which requires both computational tools and the use protocols. Protocols are inherently communicative; they describe methods to help others understand and reproduce work that has already been done. Troubleshooting and debugging are often described by the need to systematically test a problem and solve it efficiently. While debugging is often described as a form of model-based reasoning, Weintrop et al. also mention that "reproduction" of errors is a part of this practice. The reproduction of an error could imply that when users of a code experience errors, they need to be able to communicate these with their team. Weintrop et al. does not include other communication practices such as code commenting, documentation, or sharing code through repositories.

In a physics context, Gambrell and Brewe (2024) have inclusion of social practices within the computational themes and topics they identify as important in the physics curriculum. Through interviews with physics faculty and physics alumni in industry, Gambrell and Brewe found visualization is a major theme for technical skills and that justification (in terms of explaining the validity of program results, like designing models and communicating system information) was a major topic for physics and programming. Gambrell and Brewe highlight three best practices for programming: commenting, meaningful variable names, and intelligible code.

These communicative practices are not unique to physics, but rather make up a common set of best practices for computation (Wilson et al., 2014).

Computational Literacy

Another lens into computation is CL, which in contrast to CT incorporates communication explicitly (diSessa, 2001). In CL, diSessa (2001) notes that modern society is dependent on computation, and he describes computation as a new form of literacy (similar to reading and writing) through which our society interacts. The three pillars of CL are material CL, which describes the ability to use and understand a computational tool, cognitive CL, which describes the ability to apply a computational tool within the context a particular problem, and social CL (SCL), which describes how the use of a computational tool is situated within the larger community. This framework can be used to categorize the CT practices mentioned above (to code up a computational model, the syntax needed relies on material CL; the abstraction for the model relies on cognitive CL; producing visuals and descriptions requires SCL etc.). In his book, diSessa highlights the social pillar, in particular referencing the deeply social nature of any literacy. Some more specific aspects of SCL include communication of or about computation, as well as the documentation and description of code (e.g., forums, code sharing repositories, and visuals).

Computational literacy was expanded in a physics education context by analyzing the knowledge, practices, and beliefs associated with each individual pillar (Odden et al., 2019). The knowledge associated with SCL includes the strategies and tools associated with communication. In practice, communication is achieved through SCL with things like code organization, explanation, and the creation of visuals. The beliefs regarding SCL refer to the beliefs about presentation of code and presentation of the analyses. Together, the knowledge, practices, and beliefs associated with SCL provide a more detailed framework assess how students enact their CL (Odden et al., 2019). Odden and collaborators used computational essays in introductory physics, collected via Jupyter notebooks, to assess student CL. Computational essays combine textual explanations with computer code and visual outputs to describe an idea or answer a question. This student artifact allows instructors to gain insight on how students think about the elements of SCL directly by combining a scientific report with code. Computational essays have been a useful tool to investigate the development of CL (Odden & Burk, 2020; Odden & Malthe-Sørenssen, 2021; Odden et al., 2019).

Specific manifestations of CL will vary by discipline, so interpretation of SCL across discipline requires an understanding of the use of computation in those fields. For instance, the work from DeJarnette et al. (2022) highlights differences in computation between biology and engineering labs. Recognizing the differences in how disciplines interact with computational outputs and how those disciplines will contextually apply computation to solve their problems alters how that discipline is expected to communicate. In their editorial, Li et al. (2020b) describe these considerations as discipline specific aspects of computation. Despite the effects of

disciplinary context on the three pillars of CL, certain aspects can be more broadly applied to STEM with careful consideration.

Research as an Activity

In order to advance our understanding of how social and communicative aspects integrate with other computational practices, we use a socio-cultural perspective known as Activity Theory (AT) (Yamagata-Lynch, 2010). AT has broadly been used as a lens to understand learning in complex, real-world situations involving tools and communities (Engeström, 1987), such as human computer interactions (Nardi, 1996) and physics laboratories used for teaching and research (Zohrabi Alaee & Zwickl, 2023; Zwickl et al., 2023). Recent work in K-12 science education has also utilized AT to develop a framework for integrating computational thinking into science classes (Hurt et al., 2023). Activities are inherently goal oriented (Vygotsky & Cole, 1978). Individuals will have goals, and in the simplest description of AT, individuals will utilize mediating tools to achieve these goals. Following from Vygotsky's work, Engeström (1987, 2000) expanded AT to include not only the tools, but the rules (or norms), community, and division of labor individuals will experience to achieve their goal. Engeström's extension to AT created a means to analyze how a group with similar goals affect how the individual will achieve their goal. The elements of AT are often organized into an activity system (AS), which sometimes include a set of outcomes which stem from the work done to achieve a goal. Descriptions of each of the elements of an AS are shown in Fig. 1. In addition to the elements of an AS, tensions are an additional consideration that helps to investigate an activity. Tensions refer to incompatibilities that arise due to conflicts between activities or their components that disrupt the system. Tensions indicate places in the AS where there is potential for change and development to better meet the activity's goal.

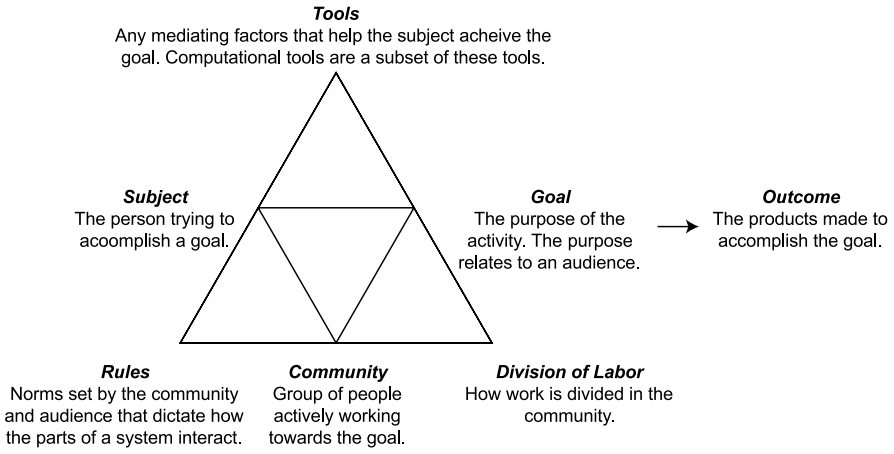

Fig. 1 General activity system. Engeström's representation of an activity system with descriptions of the elements

In this study, we refer to computational tools and an audience for research, which are separate from the tools and community described above. Computational tools are a subset of the many mediating tools that could contribute to an AS (organizational schemes, representational forms, physical equipment, learning resources, etc.). In addition to the dual meaning of "tools," we note that our study frequently uses the term "audience" (see the Goal in Fig. 1), which is distinct from the community aspect of an AS (bottom of Fig. 1). Our use of "audience" in this manuscript represents the parties affected by research goals (i.e., consumers of research and users of tools).

Methods

Interviews

We conducted semi-structured interviews over Zoom with 15 STEM faculty spanning two large northeastern and one southeastern university (Table 1). Because our interviews focused on how STEM faculty uses computation within their research, we sought faculty whose work involved some form of computation. Interviewees were identified by reviewing public websites of researchers and via recommendation by other study participants. Invitations were sent via email. Respondents came from mathematics, physics, biology, chemistry, and engineering departments. To protect our participants' identity, the table only provides their field of study, a range describing how long they have been doing STEM research and a broad description of how

Table 1 Participant backgrounds. Each participant reported demographic information via Google Form except for the last column. That column was assigned using the participants' description of their research

Pseudonym	Gender	Research experience (years)	Field	Use of computation in research
Dr. Hauser	Male	10–19	Bio	Data Analysis
Dr. Barry	Male	40–49	Bio	Modeling and Data Analysis
Dr. Berton	Male	40–49	Chem	Modeling
Dr. Coy	Male	10–19	Chem	Data Analysis
Dr. Schulz	Female	0–9	Eng	Data Analysis
Dr. Schmidt	Female	10–19	Eng	Modeling and Data Analysis
Dr. Fritz	Female	20–29	Eng	Data Analysis
Dr. Holmes	Female	20–29	Math	Modeling and Data Analysis
Dr. Ball	Male	20–29	Math	Data Analysis
Dr. Hudgins	Female	20–29	Math	Data Analysis
Dr. Atkins	Male	30–39	Math	Data Analysis
Dr. Doyle	Female	0–9	Phys	Modeling and Data Analysis
Dr. Graham	Male	10–19	Phys	Data Analysis
Dr. Banks	Male	10–19	Phys	Modeling
Dr. Hugo	Male	40–49	Phys	Modeling and Data Analysis

they use computation in their research (self-reported). Some participants included the time they spent as a graduate student or postdoc in their research experience. Additionally, we added a broad description of computational application in the "Use of Computation in Research" column. From descriptions of our participants' projects, we assigned each person either *Data Analysis* (mention of statistical tests, curve fitting, image manipulation, experimental data, etc.), and *Modeling* (mention of models, simulation, etc.), or both.

Each interview was about one hour long and focused on the discussion of computation within the context of STEM research. During interviews, participants were first asked to describe their use of computation, as it pertained to either teaching or research activities. This work focuses on researchers who discussed computation as it pertained to their research. Researchers were then asked to share their motivations to use computation, the specific tools/software/programming languages they used, thought processes needed to incorporate computation in their research, and about collaboration in their work. In addition to discussion of computation in their work, participants were asked to share computational artifacts (code, documentation, repositories, images, or other outputs) via Zoom screen share. Many researchers shared code repositories, figures they generated, and published research articles. Figures shared with us included representations of models and datasets from various computational tools. Other artifacts shared with us include data files and data structures that researchers would load into their computational tools. If at any point participants asked what interviewers meant by computational work, interviewers responded with "usage of any type of computational software" to solve a research problem, and gave a few examples, such as R, Python, and Excel. The full interview protocol is available in the supplemental materials.

Interviews were transcribed using Otter.ai and edited for grammar and correctness. Interviews were then coded using Dedoose qualitative data analysis software. For each participant, artifacts were saved using screen captures from the Zoom interview and compiled into a unique document with a timestamp before being loaded into Dedoose.

Analysis

To answer our RQs, we used an emergent coding scheme to identify the themes our participants described. We approached the analysis through the lenses of Activity Theory (Engeström, 1987; Vygotsky & Cole, 1978) and Computational Literacy (diSessa, 2001; Odden et al., 2019). This section provides an overview of our four-step analysis process, which is summarized in Fig. 2.

In step 1 of our analysis, we identified our participants' broad research practices. We parsed the interviews for phrases that started with: "What I/we/you try to do…", "I/we/you need to…". The following prompts yielded the richest descriptions of research practices: "What do you study in your research?" and "How do you use computation in your work?" In response to these prompts, our participants described

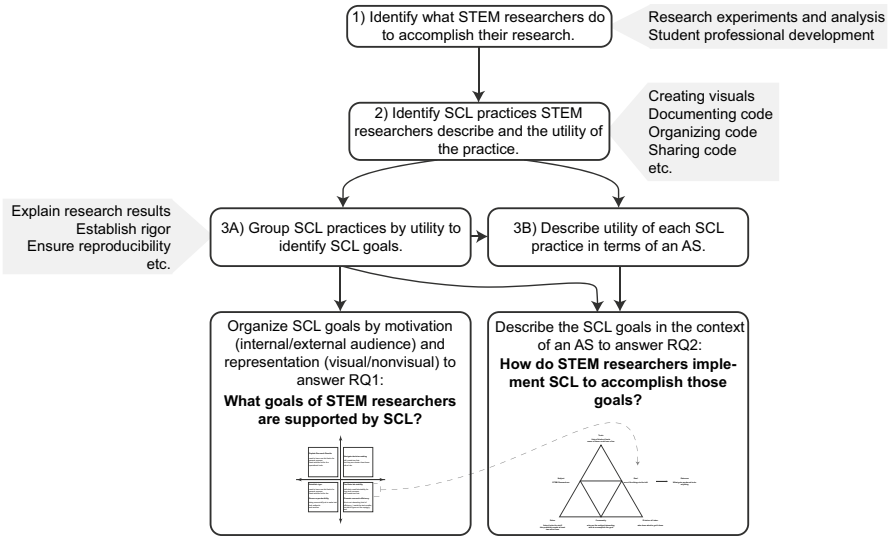

Fig. 2 Qualitative analysis flowchart. This flowchart details the data analysis used in this paper. Briefly, STEM research practices related to SCL are identified and interpreted in terms of their use in research. These practices are interpreted as goals for activities and thematically grouped and organized by motivation. These goals are further explained as an activity system, as illustrated by the dashed line connecting the goals of the framework to the goal of an activity system at the bottom of Fig. 2

their motivations in their field of study including use of computation, and needing to provide professional development for their students. The scientific research practices we identified were addressing specific aims for their projects, writing papers, and synthesizing computational and experimental projects. The practices we identified related to student professional development were training students to contribute to the project, preparing students for their careers, and giving students the chance to present their work in lab meetings and at conferences.

In step 2 of our analysis, we identified the set of practices that pertained to SCL, and how those practices were employed. To identify SCL practices, we focused our analysis on (a) discussions about participation in a computational research collaboration, (b) communication with computational tools (e.g., documentation and organization), (c) communication through computation (e.g., the creation of visuals), or (d) collaboration and communication about computational tools (e.g., code sharing, tracking version history, and pseudocoding). Coupled with the fine-grained research descriptions, our participants' discussions of SCL practices were usually in response to the following prompts: "What documentation practices do you consider best?," "How do collaborations work for you?," and "Are there any non-coding activities you do that require computational thinking?" Many participants described practices related to planning and organization within the research lab in response to the non-coding question. We analyzed interviews one at a time, adding to the list of practices with each successive

interview. The list of practices reached saturation (i.e., no new practices were being added to the list) after 8 interviews.

In step 3 of our analysis, we analyzed the social practices identified in step 2 as goal directed activities in alignment with Activity Theory. Step 3 A of this analysis involved grouping the practices identified in step 2 by motivating factor or into categories of goals. For example, we found that organizing code was an important task that our participants mentioned. They mentioned that having "legible" code, having plans for how they would write their code, and being able to understand their code documentation were reasons they needed to organize code. Contextually our participants discussed this organization in terms of code being useful to new members of projects, so we grouped these SCL practices into a goal called *facilitate lab stability*. Similarly, we created categories for the following goals regarding the SCL practices that our participants mentioned: *explain research results*, *establish rigor*, *ensure reproducibility*, *understand data*, *facilitate lab stability*, and *promote research efficiency*. We then analyzed all interviews, coding specifically for mention of any of those SCL goals. Step 3B of our analysis involved generating AS's for the SCL goals determined in Step 3 A using all interviews to describe the different elements. Our participants mentioned using specific tools when discussing each practice and commented on the community and division of labor but often mentioned norms indirectly. We organized the SCL goals by motivation and representation to address RQ1, which is explained in detail in the "Results" section. Furthermore, we compiled the AS's created in step 3B to generate contextual descriptions of each SCL goal to address RQ2. These AS's are presented in detail in the results section.

Results

One differentiating feature between goals was the audience for SCL activity. For STEM researchers, this audience could represent other researchers in the field, collaborators at other universities, collaborators at their home institution, or their own research group. We separated the SCL goals based on whether the activity was focused on researchers external or internal to their network of collaborators (horizontal axis of Fig. 3). The motivation for these goals is intimately tied to the intended audience. The external audience goals are persuasive in nature because they focused on convincing others in the scientific community about the quality of the research. Internal audience goals focused on helping the internal function of the research group to be more productive. The goals were also differentiated based on the computational and representational tools used to accomplish them (vertical axis of Fig. 3). There are goals achieved primarily through visual representations and goals achieved through nonvisual representations.

The six SCL goals (*explain research results*, *establish rigor*, *ensure reproducibility*, *navigate decision-making*, *facilitate lab stability*, *and promote research efficiency*) emerged as distinct reasons that researchers use visuals, documentation, organization, and repositories in their work. The next few sections present how our participants accomplished the SCL goals, organized by quadrant of the framework (Fig. 3). To

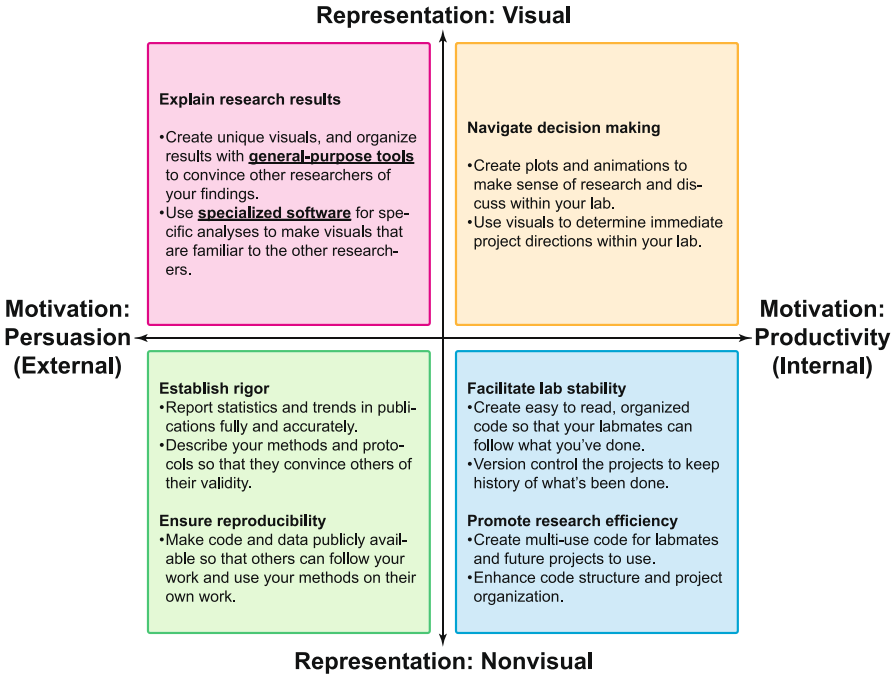

Fig. 3 Framework to interpret the goals SCL play in STEM Research. SCL is used either persuasively or for productivity (horizontal axis). To accomplish this use, SCL is implemented either visually or non-visually (vertical axis)

describe the motivation and representations, we use hyphenated nomenclature (e.g., visual-persuasion) and continue to italicize the goals identified above.

Persuasion (Visual) Goals of SCL

Starting with visual-persuasion in the top left quadrant of Fig. 3, 14 of our participants mention creating visuals as an essential use of computation in their research. Dr. Doyle described computation as a powerful tool to aid scientific communication because "it's tactile, it's visual, it's things that you can change really quickly." From our interviews, we identified an AS for visual-persuasion (Fig. 4) that describes some of the common aspects from our interviews. In the following section, we describe our finding that two partially distinct AS's emerged across participants who used general-purpose tools versus more specialized tools to create visuals. Of the 14 participants that mentioned visual creation, 12 described their use of general-purpose tools, while 7 mentioned their use of specialized tools. We start by describing their commonalities between visual creation with general-purpose and specialized tools before describing their unique aspects.

The *goal* of the visual-persuasion activity is to *explain research results* to others convincingly. For example, Dr. Banks described how useful animations can be

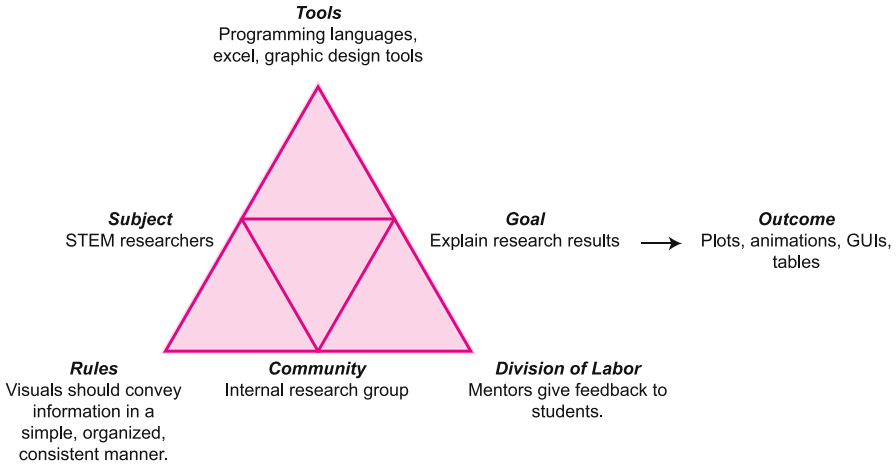

Fig. 4 Activity diagram for the visual-persuasion goals. The SCL goal being addressed by visual-persuasion is explaining research results

to present research talks: "[Animation] makes fun viewing… and if I'm presenting talks or my students are presenting talks, it makes [the science] easy to explain sometimes." In addition to the making work easier to understand, Dr. Fritz highlighted this goal by referencing the value researchers place on visuals: "When you ask about why do we use [a scanning electron microscope and image analysis], that one is very, very critical for research, if people don't see it, if they don't see a picture, they don't take it for value." Furthermore, our participants mentioned that visuals can help students to think about the audience. Dr. Schmidt encouraged her students to "focus on their stakeholder interactions," and be creative with the use of their animations to help students "understand what someone may value at the end of the day looking at their output."

The *outcomes* our participants mentioned for this goal were images, figures, plots, GUIs for developed tools, animations, and tables. Dr. Schulz has used visual elements such as "summary tables, charts and graphs… in our publications." Dr. Schulz described how those products went "beyond [her] tiny research group" to share with others.

In contrast with the audience for persuasive goals, the *community* through which these goals are mediated can be complex. There was no mention of the roles different researchers had in developing visuals for persuasion, but rather the entire research group had the same goal and worked in similar ways to achieve it. Mentoring was mentioned by many participants in discussing their research, and creating visuals was often learned through an iterative process with feedback provided by the mentor.

General-Purpose Tools

The general-purpose *tools* that support visual-persuasion are the programming environments available to researchers and the features of those programming environments. Software such as MATLAB, Python, Mathematica, Java, Fortran, R, ImageJ, and Excel allow researchers from various disciplines to do their own data manipulations and create publication quality figures. Researchers also mentioned the ability to display their creativity with general-purpose tools. Dr. Banks' student learned a single language (Python in this example) and from there use their "crazy imagination… She's making us different structures, videos for all the simulations." Experienced users can finely tune their graphics when plotting functions to convey their stories accurately and convincingly. For example, Dr. Hugo showed a figure that he and his team made in Mathematica (Fig. 5). Dr. Hugo used color to represent the energy associated with different interactions between protein charge patterns. Energy was plotted with respect to a specific two-dimensional slice through a six-dimensional space (Fig. 5). Additionally, our participants frequently mentioned the use of tables in publications as an organizational *tool*. For example, Dr. Schulz measured the effectiveness of an in-class intervention and reported the results in the table to make quick comparisons

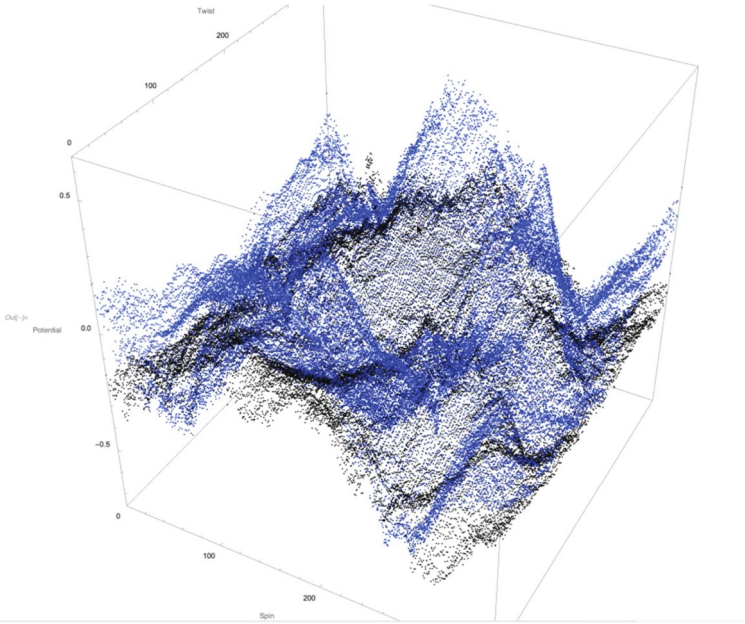

Fig. 5 3D plot made with a general-purpose tool. Mathematica was used to visualize an output from a complex parameter space. Both color and the axes are adjusted by the coder to display a specific result

between her control and test groups. Summary tables supplement and at times even replace sections of text in publications.

The disciplinary community imparts a set of *rules* that researchers must consider when trying to *explain their research results.* Because general-purpose tools can be used across many communities, we found that researchers must actively incorporate rules regarding the presentation of figures for their field. Dr. Barry mentioned needing to use visuals for clarity so that "there might be a lot of graphics, and then a… small number of words." Furthermore, in Dr. Hauser's research, figures often "have uniform ways of presenting similar variables," for organization, to *display* competence, and reduce the cognitive load of the audience.

Specialized Tools

Another way STEM researchers in this study achieved visual-persuasion is by using discipline specific tools, which we refer to as specialized tools. These tools allow researchers (especially researchers with minimal coding experience) to create highly specialized figures from complex analyses and simulations with relative ease. These tools have a more limited applicability but may become widely used as they empower a disciplinary community to engage in particular forms of analysis and visual-persuasion. Specialized tools can be an access point for researchers to engage in computational analysis without experience scripting or traditional programming. As with the general-purpose tools, we saw that the rules and tools were distinct features but also found a unique role for the division of labor within this system.

The specialized *tools* that our participants mentioned were built by other researchers in their research field. One example of a specialized tool is TBtools (Chen et al., 2020). TBtools is a toolkit used by biologists to help analyze large amounts of bioinformatics data. In addition to his work with general-purpose tools, Dr. Hauser described TBtools as "a software that allows you to construct [plots] more easily for non-coders… and since [it was published] it's kind of hijacked the field, whenever people do this type of analysis, they're usually using this tool." Since being published in *Molecular Plant* in 2020, TBtools has been cited almost 8000 times. This tool was built to allow more individuals to interact with big data (such as gene sequencing data) graphical representations of big data. TBtools also provides a graphical user interface (GUI) with example inputs for users as opposed to a command line or writing script files (Fig. 6). Many other specialized tools were mentioned by our participants, such as CrystalMaker (Palmer, 2015), TensorFlow (Abadi et al., 2016), PyTorch (Paszke et al., 2019), and LAMMPS (Thompson et al., 2022). Some of these tools utilize GUIs to support the user, while others consist of specialized packages built within a general-purpose tool, for instance LAMMPS was originally released as open-source code based on Fortran and has been ported to C, C++, and Python.

Specialized tools are often built with discipline specific norms in mind. As such, *rules* associated with visual creation can be built into specialized tools. For example, TBtools has automated the creation of commonly used visualizations, such as expression data with *eFP Browser*, *Interactive Heatmaps*, and more (Chen et al., 2020). The labeling, positioning, plot shapes, and color usages are all built into the

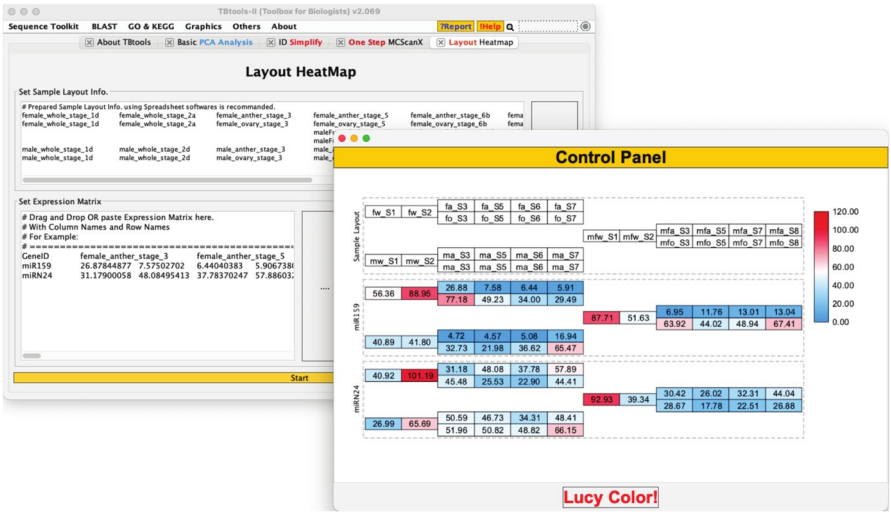

Fig. 6 GUI for TBtools. The GUI for TBtools allows users to quickly create visuals, and provides users with sample data

analysis, so users can create publication quality data. Use of the TBtools GUI allows the researcher to generate a figure that follows all disciplinary norms (Fig. 6). For the specialized tools, aesthetic and discipline-specific rules were not mentioned, and this is likely because the tool incorporates these rules automatically.

The use of specialized tools helped to reveal an aspect of the *division of labor* within visual-persuasion. Some individuals within the broader research field are specialized tool makers, that is, they create the software tools that are shared for others within the field to use. Most researchers use these tools but are not actively developing them.

Persuasion (Nonvisual) Goals of SCL

Nonvisual-persuasion goals make up the lower left quadrant of Fig. 3. Of the 15 participants, 8 mentioned participating in nonvisual-persuasion goals; 5 of those 8 participants referred to supporting analyses textually, while the other 3 described sharing their code for the purpose of convincing others of the quality of their work. Similar to visual persuasion, our participants who engaged in nonvisual-persuasion attempted to persuade external researchers and reviewers (e.g., manuscript referees and conference attendees). Here, we present one AS that supports two nonvisual-persuasion goals that we identified: *establishing rigor* and *ensuring reproducibility*. *Establishing rigor* through textual explanations and descriptions of research and *ensuring reproducibility* by creating code repositories both aid researchers to convince others of the validity of their work.

Researchers described accomplishing both goals using individualistic means so the AS (Fig. 7) does not have significant description under *community* other than the internal

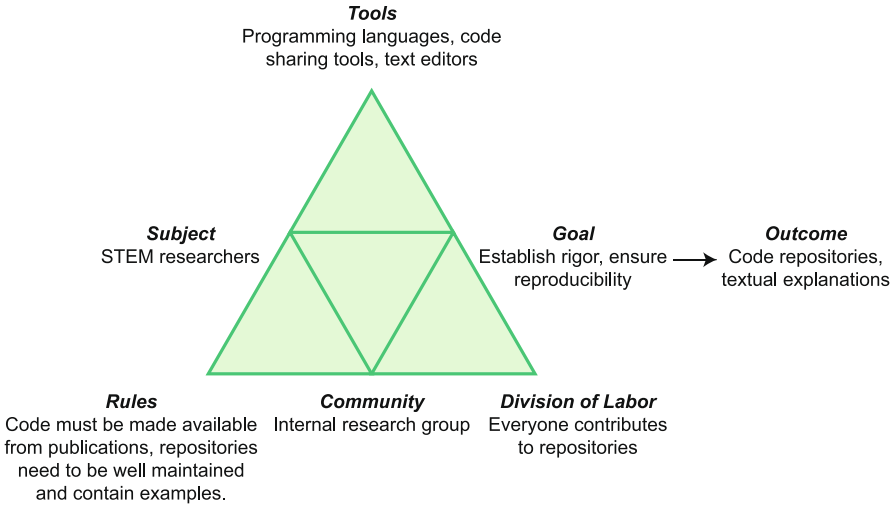

Fig. 7 Activity diagram for the nonvisual-persuasion goals. The SCL goals being addressed by nonvisual-persuasion are establishing rigor and ensuring reproducibility

research group, or *division of labor*. The internal research group shares the burden of creating and maintaining their code and descriptions of that code.

Establishing Rigor

The *goal* of *establishing rigor* is to convince others that your methods are valid. Our participants described research quality and validity by defending all aspects of their work. Rigor involved writing statements about how other researchers can access data/code for the purpose of replication or reanalysis, as well as describing methods for algorithm implementation and verification. For example, Dr. Atkins described the need for a simple, explainable analysis method with a counterexample: "[What is] important in data analysis is… attention to detail… trying not to do the same thing in five slightly different ways. Because then when you have to explain it to somebody afterwards… there's so many exceptions to the description of what you said you did." In Dr. Atkins' telling, a lack of consistency could have led to difficulties in defending his chosen method.

Outcomes associated with *establishing rigor* in research are primarily textual. Explanations in scientific papers are made to satisfy the need to convince others of rigor. For example, Dr. Fritz reported that she shares engineering methods she develops for 3D printing because "it's a brand-new process, and nobody understands [what you have done], you provide information about how that process works. It could be parameters… and [a protocol] gets you this type of surface finish, for example."

The *tools* researchers mentioned using to mediate nonvisual-persuasion included the general-purpose tools that researchers use to create visuals, as well as text editors such as Microsoft Word, Google Docs, and the markdown sections of Jupyter notebooks.

For ensuring rigor in a simulation, one example of the textual description came from Dr. Barry. When creating "a still image of two molecules interacting," Dr. Barry made sure they were "annotated [so] you know this carbon's interacting with this nitrogen, for example… with an arrow and a little in words." In addition, those visuals required "a whole lot of words" to describe what was done and how it was done because the visual alone was insufficient.

There are certain community norms (i.e., *rules*) that researchers abide by for nonvisual-persuasion. Our participants discussed the importance of being able to describe research results. When faced with highly variable data, Dr. Coy was unable to move forward with a project until he could defend the results. Dr. Coy said that "the correlation, statistically, [is] not so good. So it tells me… we've got to tighten up our manufacturing because the model is pretty useless at the moment because… I can't trust it at all." Even though the model produced a result, it was not good enough to publish.

Ensure Reproducibility

The *goal* of *ensuring reproducibility* is sharing code and data with the greater audience. Dr. Ball's code in MATLAB Central had over 2200 downloads, which other researchers used to verify Dr. Ball's results and analyze their own data. Dr. Ball reported that his repositories were also "on GitHub, so that new researchers can download it. And they can run the data… through [the code] to make sure that they're convinced that I was… ethical in what I published, but they can also run their own data through it." Researchers create repositories with these uses in mind, and journals and funding agencies frequently require or encourage availability of code and data for this reason as well.

The *outcome* for nonvisual-persuasion is the code repository that shares both the code and the computational outputs associated with your work. In building his repository, Dr. Ball recalled that he published "a conference paper… where we described the [image analysis] algorithm and we showed how it worked on various images… and then at the end of the paper, if you go look in the published paper, it'll say, 'prototype implementations are available for download at MATLAB Central.'".

Specific code sharing *tools*, such as GitHub, MATLAB File Exchange, and JavaDocs, are used to allow other researchers to have access to code and data. Dr. Graham mentioned that example documentation and code usage guidance within a repository was a tool to help researchers codeshare. Dr. Graham showed us his documentation and described how Jupyter notebooks are used to weave descriptive text in with example code snippets. The documentation Dr. Graham showed us is "accessible from GitHub. There's tutorials. And in the examples, there's lots of different things. So if you say I need to do X, Y, or Z … you can actually see how the cells run."

When it comes to accessing work through repositories, our participants described important implementation *rules*. Going back to Dr. Graham's repository, he described that his group does "most of our code in… scripts, but [also uses] notebooks, because it makes it easier… for other people to see. So… we go through step by step… 'we're going to add this component, we're going to add that component.'" The entire group is

expected to contribute to Dr. Graham's repository in a similar way (e.g., the same language and format) to make sure each step in the process is well documented.

Productivity (Visual) Goals of SCL

Visual-productivity (quadrant 1 in Fig. 3) marks a shift in audience from external researchers to the internal research team. Researchers mentioned that one goal of SCL for them was to *navigate decision making* with visuals. Ten of our participants discussed their use of visuals during decision-making processes within their research. This activity shares the tools and outcomes discussed in the visual-persuasion. Here, we present one AS that encompasses the uses our participants mentioned for visuals that help to progress their research (Fig. 8).

The *goal* of visual-productivity is to use visuals to help the decision-making process in STEM research. Researchers mentioned the need to understand their simulations so that they can interpret results and new directions, as well as understand existing experimental data to inform their simulations. Part of the decision-making process involved internalizing research ideas and embedding them within the needs of a research field. The use of visuals helped for research ideas to "enter [Dr. Hugo's] consciousness in a more direct way". Dr. Hugo saved time and solved a problem related to the free energy of a system because a "picture made for an idea. Because I found the picture. I said, 'Hey, wait a second. Now, that's a convex surface. It looks like a convex surface.'" Similarly, decision-making involved finding intermediate results for model creation. Dr. Doyle stated that she has to "[plot] experimental data and do some analysis to figure out what [the summary] parameters are … [so they] can then serve as input[s] to my models" so that her biophysical model could more accurately recapitulate a biological system. Our participants mentioned the need to train students to work productively, which

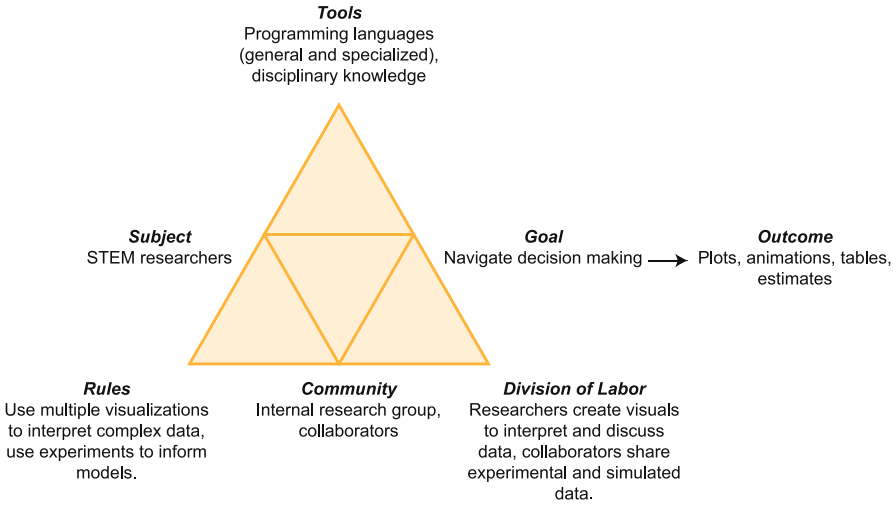

Fig. 8 Activity diagram for the visual-productivity goals. The SCL goal being addressed by visual-productivity is navigating decision making

included the creation and understanding of visuals and the development of an ability to use those visuals to make research decisions.

The *outcomes* for this activity are plots, graphs, animations, tables, and the decisions researchers made regarding their work. Dr. Coy mentioned using animations and 3D constructions with his machine learning algorithm. He explained that when "we have 1000 different molecules… and you've asked a machine learning algorithm to pick the simplest molecule with the simplest manufacturing… you look at the top ten, and you … look at what's actually been [identified] based on your model." Dr. Coy used the visuals created in this process to decide which output of the algorithm was relevant to the process he was studying.

Our participants used both general-purpose and specialized *tools* to create visuals for this goal. To make a decision about the best measurement healthcare workers should make during a cardio test, Dr. Holmes used MATLAB. By simulating the outputs of a model heart and using "codes that generate visual aids… that are representations of model properties," Dr. Holmes explored what "the most informative measurement" would be for healthcare workers to make. This reevaluation step allowed Dr. Holmes to change research directions and incorporate new ideas into her model. Dr. Berton used specialized software called Moltimate to compare protein structures in his research. Dr. Berton stated, "[Moltimate] does a search to find the alignments for that structure… [and compiles] data on the right-hand side here that tells us which amino acids… align well with… a protein of known function." Dr. Berton then used information from that analysis to better develop his research tool.

The *rules* pertaining to visual-productivity referred to the need to trust the decisions that were made. Among our study participants, researchers generally used an iterative process and double checked their results. Dr. Schmidt declared that when she evaluates her simulation modeling, she has to "analyze the output, and… say 'Does this output represent what we should expect?' And if it doesn't, then that gives us a sense of understanding that we need to go back and maybe figure out where it went wrong." By using this evaluation process, Dr. Schmidt ensured that she trusted her decision enough to continue. Similarly, when Dr. Doyle solves for parameters for her biological models, she uses multiple means to reach a conclusion, suggesting discipline specific norms as well. By using two methods to estimate a model parameter, Dr. Doyle recalled asking herself "Do I get the same value through this estimation versus the curve fitting or not? And how far [are they] from each other?".

Researchers, students, and collaborators make up the *community* that participates in the decision-making process. Researchers reported that they and their students would work on analyses and make decisions independently and then come together to talk. To accomplish this, study participants focused on training students. For example, when Dr. Schulz talked about visual-productivity, she emphasized the need for herself and the experienced students to help the inexperienced students to explore various representations so they could "learn on the job." Some of our participants also mentioned that collaborators help them to make decisions for their projects. Dr. Hudgins' group met with her collaborators to discuss "what [they] have seen" and "talk through… the meaning of [her work]," as well as ask "some questions [she] had."

The *division of labor* we identified for researchers in this activity was centered on mentor–mentee relationships. As for the teaching aspects of visual-productivity, Dr. Doyle described needing to gauge the level of understanding students have with figures and analyses. Dr. Doyle "asked one of [her] students to… find the slope of the decay, and what is the distribution in the slope… because every time you do this … there's a little spread." In this task, Dr. Doyle wants the student to "figure out how to find the slope and the distribution of the slopes, [because] there is no specialized software for it, you… have to figure out how to do it yourself." By allowing a student to take this approach, Dr. Doyle tried to get her student to a more independent level. Aside from mentorship, the community for visual-productivity all undertake the role of "decision-maker" so that when they discuss the decisions and reach a conclusion, the entire group trusts the decision.

Productivity (Nonvisual) Goals of SCL

The final section of this framework is nonvisual-productivity (quadrant 4 of Fig. 3). Similar to visual-productivity, the audience and community for this activity are the internal research group. Similar to nonvisual-persuasion, the primary outcomes were code repositories and comments. Here, we present one diagram that describes non-visual-productivity activities in general, followed separate detailed explanations of how SCL is used to *facilitate lab stability*, and to *promote research efficiency* (Fig. 9). Almost every interviewee, 14, mentioned how SCL *facilitated lab stability*, and all 15 mentioned how it *promoted research efficiency*. This was the most prevalent utilization of SCL in our interviews. In this section, we discuss the tools, community, and division

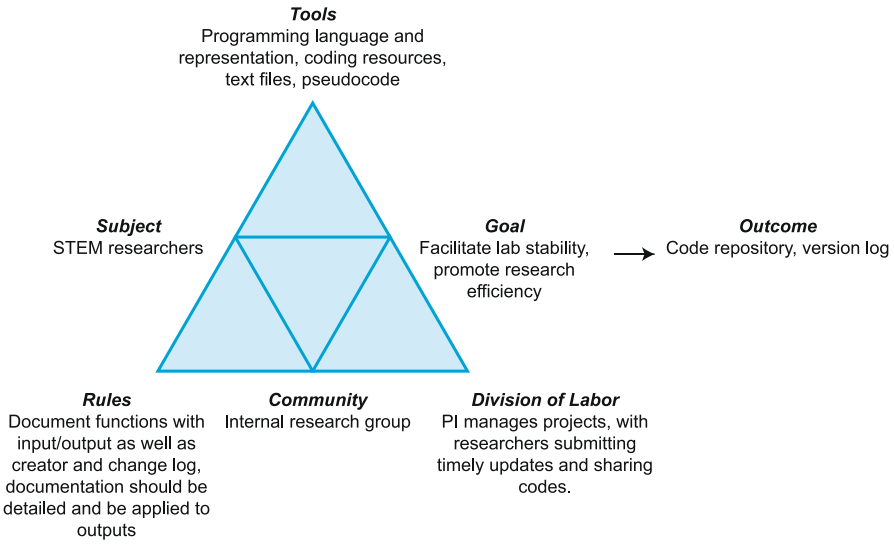

Fig. 9 Activity diagram for the nonvisual-productivity goals. The SCL goals being addressed by nonvisual-productivity are facilitating lab stability and promoting research efficiency

of labor that are common to nonvisual-productivity goals, then separately describe the goals, outcomes, and norms for *facilitating lab stability* and *promoting research efficiency.*

The primary *tools* that were used for nonvisual-productivity goals were general-purpose coding tools, code sharing sites (e.g., GitHub), and text documents (e.g., Microsoft Word). Dr. Holmes described how a lab notebook in Word allows researchers to keep a "diary… [with the] codes you're running… [and] some of the outputs." She went on to say that a diary allows researchers to put together the comprehensive version logs and build a code repository because it gives them a "running commentary" that lets them see "where [they] left off." Another tool used by our participants to aid nonvisual-productivity through SCL is pseudocode. Dr. Barry remarked that his group creates a "flow diagram of what [researchers] are going to do" to think through biological network identification "logically and algorithmically."

The *community* that participates in nonvisual-productivity differs slightly from the visual-productivity quadrant. There was no mention of collaboration when discussing nonvisual-productivity. The internal research group was the focus of researchers discussing these goals.

In addition to the student-mentor interactions we saw in the visual-productivity section, students took on varied roles. For example, Dr. Hauser' describes a *division of labor* within his research group where some students "[test] the effect of disease or assault or stress or just hormones on [the] lipids [he] works with, [while] other students do more computational work… [on] the expression of [those] genes." Even the roles within a computational group of students can diverge. Dr. Berton has had students who are "assigned to try and break the code" by running "extreme versions of… 'what somebody might do.'" The "codebreakers" will communicate error messages to the coders so that they can improve the code.

Facilitate Lab Stability

Lab stability *goals* through SCL include code organization and version controlling to make sure individual contributions are documented. One concern our participants mentioned was a constantly changing research group due to older students graduating and newer students joining. New students have to use code generated by previous students. Dr. Barry compared stability goals in STEM to programming in industry: "Every line of code has a page of documentation. So that five years from now, when the company is supporting that software product and the person who wrote it is long gone… they know what each line was intended to do." This same goal focuses on making sure a researcher understands their code as time passes. Dr. Barry went on to say that researchers document this way so that "six months from now… you know what you did."

The *outcomes* of this goal are related to the created code itself. One of the outcomes of *facilitating lab stability* is a well-documented code base with a version control system. Dr. Ball emphasized that "if you're working with more than just you, you want a version control system that allows you to check in an initial version of code. And then if people are making edits, they can check out a version, they can

check it back into logs, and [track] all the changes that [they've] made and… go back and undo changes really easily." Other researchers use less formal version controlling such as a text document that describes exactly how analyses used in publications were done. Other outcomes focus on the readability of code. Dr. Hugo acknowledged that he has a "habit of using very long function names," which can include the day a function was written. Whereas Dr. Hugo uses a naming system for version control, Dr. Atkins used descriptive variable names for data types and variables so that they could be interpreted quickly and easily, and made it so the "pseudocode doesn't look that different from the actual code."

We noticed some of the *rules* for this activity from the examples above. Dr. Graham mentioned that commit messages need to be clear and descriptive enough to know what changes were made and why. Dr. Atkins wanted his students to code in a way that their finished product looks like pseudocode. In addition to these examples, Dr. Doyle commented that she makes sure her students "write notes so that whenever [their mentors] have to use their code, [they] know exactly what [the students] are doing." This form of documentation is the foundation of keeping a version history and change log for projects. Similar documentation and rules are expected during code creation and organization. Dr. Barry said that documentation and code must "be written clearly enough" so that both the person who wrote the code as well as whoever will read the code can understand it. Dr. Barry also emphasized the importance of "[avoiding] disciplinary jargon especially" in his documentation.

Promote Research Efficiency

The other nonvisual-productivity *goal* we noticed was promoting research efficiency. Many of the tools, as well as the community, are shared between both goals. However, we found the norms to vary. Research efficiency is supported by things like project organization, problem decomposition, and creation of multiuse code. Dr. Hudgins pointed out that without organization, researchers might "get excited about [something] in the data set, and… start running things, [but]… realize [they had] already done that." By organizing the project in this manner, Dr. Hudgins ensures her lab will not duplicate work. Dr. Coy described the process as "[breaking] down tasks into clusters, and… [writing a] problem out into smaller parts and [tackling them]," and jokes that this task is analogous to the question: "How do you eat an elephant? One bite at a time." Here, Dr. Coy uses problem composition as an organizational tool to write out the tasks required to solve his problem. Another goal described in detail in the next paragraph focuses on repeatability in research. The same way repositories are used with a focus on version controlling for lab stability, repositories and template codes are used, and reused to *promote research efficiency*.

The *tools* our participants mentioned using included organizational tools, such as Microsoft Word, or even pen and paper, as well as the code bases containing the generalizable codes researchers created. Creating "templates" with modularized code, and repurposing tools within a code was mentioned by five of our interviewees. Dr. Hugo described using these versatile codes as having "tinker toys [to] play with" so that he could make functions of functions to mirror how he thinks in computing. Dr. Atkins takes this outcome one step further to code "makefiles," which are scripts that generate

lines of code to run. By using makefiles, Dr. Atkins avoids the bad practice of "commenting and uncommenting" lines of code (Wilson et al., 2017).

Some of the researchers in our study described *rules* for promoting research efficiency in their descriptions above, such as Dr. Hudgins avoiding the duplication of effort, and Dr. Atkins avoiding "commenting and uncommenting." In addition to these, a norm our participants mentioned was using existing resources from the literature. The ability to "piggyback off of [what] someone's likely done… already" using available "shareware" like GitHub should be utilized to expedite problem solving. By "shareware," Dr. Atkins referred to open-source codesharing. While code repositories are typical now, decades ago, the useful code was more likely to be found in programming books. For Dr. Graham's data analysis, the expectation is that he can add a dataset without "[starting] from scratch… [by building] a pipeline," rather than "[having] ten different graduate students… [doing] things by hand."

Tensions

One tension that disrupts the nonvisual-productivity quadrant is a conflict between lab stability and research efficiency. Dr. Atkins highlighted this tension when talking about his data files: "I think one big question is flexibility of reading versus efficiency of storage… there are file formats like HDF5 that will store the floating point numbers more efficiently and… organize the tables… but then you need a tool that is going to organize that… Whereas if you do it the dumbest way possible, and just spit out a text file with a bunch of numbers in it… that's not very efficient because you're using a lot more bytes to store each number… but … you can just open up the text file and look at it." In Dr. Atkins' example, the need to keep file size down, as well as the field norms are promoting efficient work but create a barrier to new students interacting with raw data, which negatively impacts lab stability. A similar tension arises when community norms do not support the *facilitation of lab stability*. Dr. Graham described the use of "old" files that are "understandable, but not when [they're] not explained." Dr. Graham's data contains a 13,000 line text file which, in his words, is "notoriously horrible to deal with," (Fig. 10). Dr. Graham discussed how the old files are kept for research efficiency because the data analysis pipeline is written to interact with the text files, at the cost of a simpler storage method that new users could more easily understand.

A different tension that can hinder research efficiency is the tension between using black box software versus custom built software. Two researchers mentioned making their own software even though other products are available freely or

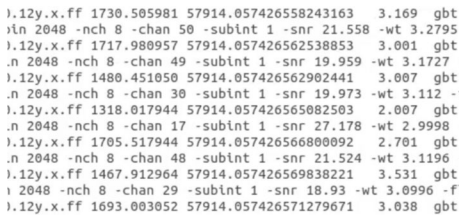

Fig. 10 Text file containing data related to Dr. Graham's research. This text file contains raw research data which is often difficult for a novice to read and understand

commercially. The reasons for developing their own software included researchers not knowing "how many bugs there might be" or an unwillingness to "[live] with the limitations" of an expensive and/or opaque tool. Researchers also mentioned a decreased level of trust in results when important analyses are hidden within a "black box" software. For that reason, some researchers opt to use more transparent or custom software, potentially at the expense of efficiency.

Limited computational resources also present challenges to these activities. A few of our participants mentioned that the files they need to load on their machine are too large for their personal computer to open. To circumvent this, researchers tried to use other means of computation like cloud computing and use of local computer clusters to do their research. This kind of change from personal machine to cluster often requires unique SCL to interface with the cluster from your machine and parallelization of your code to utilize multiple cores. Likewise, Dr. Barry faced physical constraints in his collaboration. Many of the files his team works with are too large to send via email, and Dr. Barry was weary of cloud storage for data transfer. Dr. Barry often finds the only way to collaborate is to transfer hard drives to other researchers, and as a result tends to work with data and simulations more internally, and his collaborations are limited to occasional joint group meetings.

Discussion

Our Framework

Our interviews with STEM researchers helped us to formulate a two-dimensional framework for conceptualizing SCL. Our framework interprets the practices associated with SCL in terms of researchers' motivations to implement that practice (i.e., persuasion or productivity) and by the representation used (i.e., visual or nonvisual). Separating SCL practices along these axes helped to contextualize the previous formulations of CT and describe why and when certain practices are useful (Grover & Pea, 2013; Weintrop et al., 2016; Wing, 2006). While documentation, commenting, and code organization are often mentioned as best practices in CT, the use of these practices to solve problems is often not mentioned. When they are mentioned, best practices describe why things like commenting is helpful to read code, without mention of an audience (Gambrell & Brewe, 2024). Our work builds on best practices such as commenting by describing when and why commenting matters, comments foster stability in the face of frequently changing lab members.

We identify practices that are not an obvious extension from any of the formulations of CT cited in the literature review. Whereas CT often leaves communication and collaboration as implicitly defined within the themes and practices, our work explicitly highlights the social nature of computation. Documentation for rigor is one such computational practice that STEM researchers need to accomplish in their work. Well documented computational work will instill confidence that this work is valid and trustworthy. We also identify creating visuals using specialized tools as a computational practice. This practice is separate from the use of general-purpose tools in the material knowledge the user needs to solve a problem. Specialized tools

are often built to accommodate for less experienced users and can be used by following protocols.

We extend the practice of visualization to incorporate creating visuals to aid productivity, rather than visuals primarily intended for communicating results to an external audience. CT frameworks that focus on practices often do include visualization as a key practice. Weintrop et al. (2016) focus on using visualizations for persuasive reasons, to display findings, share data, and convey information. Other descriptions of CT similarly mention visualization without a significant description of a productive element use outside of conveying information if at all (ISTE & CSTA, 2011; Juškevičienė & Dagienė, 2018; Kalelioğlu et al., 2016). Our work forefronts persuasive visualization, common to other descriptions of computation in science as well as productive visualization. By including productive visualization, our framework can help to influence more authentic computational activities.

Identifying the division of *labor* between tool makers and users in a discipline suggests an interesting interaction between the groups. Vee (2017) identifies the idea of "collective literacy" (p. 185) as it applies to computation. Collective literacy describes how group people can depend on a few literate individuals to fulfill the needs of the group. Specialized tools such as TBtools can increase the collective literacy of a disciplinary community and can help to mediate the technical skills needed to interact with large datasets and do complex analyses. Tool makers in these groups can compensate for a lack user knowledge so that everyone is "effectively literate" (Vee, 2017, p. 186). Researchers have access to a wider array of methods and analyses by relying on the tools created by the broader field.

Educational Implications

This framework serves as a guide that educators can use to help design lessons to model genuine practices in a classroom setting. Impactful CT frameworks such as Weintrop et al. (2016) and Beheshti et al. (2017) set a standard to use expert interviews to inform classroom practices. With respect to communicative aspects of computational courses, learning objectives could be aligned to the SCL goals (*explain research results*, *navigate decision-making*, *establish rigor*, *ensure reproducibility*, *facilitate lab stability*, and *promote research efficiency*).

Certain lessons can be adapted for students in courses to specifically teach elements of SCL. A general approach to enhancing SCL education is through collaborative, extended projects in the classroom. By combining and extending assignments in courses and labs, students will be able to engage in SCL in a more authentic way. A similar suggestion was made by Moskovitz and Kellogg for making more authentic science lab courses (Moskovitz & Kellogg, 2011). Moskovitz and Kellogg suggest that authentic lab activities promote authentic writing for students, and a similar engagement in computation would support the development of SCL and communication in students. Some examples of authentic course design for computation can be drawn from Software Carpentry and the Ten Simple Rules for computation series. In their works, Software Carpentry details the essential practices related to how researchers should code and suggest

social aspects like using meaningful variable names, creating multi-use functions, and tracking changes (Wilson et al., 2014, 2017). Similarly, the Ten Simple Rules series highlights the importance of robust (Taschuk & Wilson, 2017), usable (List et al., 2017), and reproducible (Sandve et al., 2013) code, and gives a roadmap for the use of GitHub (Perez-Riverol et al., 2016). Another group has created a framework to aid in activity design specifically to invoke computational thinking in the sciences (Hurt et al., 2023). This framework includes three cognitive processes (reflection about the use of a computational tool, design of a computational tool, and evaluation of a computational tool) and four authentic scientific activities (data collection, processing, modeling, and problem-solving). The framework supports the development of classroom activities that integrate one or more cognitive processes with one or more scientific activities, so educators can thoughtfully target both CT and scientific practice learning goals.

The nonvisual-productivity quadrant of the SCL framework may require significant changes to classroom activities for authentic implementation. In universities, collaborative learning can be limited to informal interactions (Mason, 2020). As mentioned above, the creation of multiweek projects could promote the need for more formal collaborative learning. Our interviewees identified GitHub as a major codesharing platform to facilitate collaborations. Explicit use of GitHub to support nonvisual productivity could broaden the learning objectives addressed in courses. For example, Zagalsky et al. (2015) interviewed users of GitHub for Education and found that undergraduates engage with the repository and use change logs, comment on the use of aspects of their codes, collaborate with one another in a class-wide forum, ask questions to the instructor, use each other's code, report error messages, etc. A separate study used a "karma" system to encourage students to interact with one another in the collaborative coding environment which received positive feedback from students and emulates authentic practice (Kilamo et al., 2012).

To further address teaching the nonvisual-productivity quadrant of this framework, real time feedback can help to support students. Add-ons to computational environments can be developed to facilitate social challenges of computation such as organizing code, documenting changes, and understanding error messages. Charles and Gwilliam (2023) recently published an automated error message feedback tool for use in Jupyter notebooks. By providing students with detailed descriptions of error messages, they were less likely to become frustrated and discouraged. The add-on to provide this feedback was created by hand for the common errors students might face, but there is potential for similar add-ons to be created with more specialized feedback through the use of LLMs. Specialized, automated, instantaneous feedback add-ons to programming environments can help the students develop better commenting practices or help track changes using plain text.

The suggestions above encompass traditional coding environments with general-purpose tools, but our results highlight the importance of discpline-specific

specialized tools (e.g., TBtools) as well. These specialized tools are not suitable for an introduction to coding (because user interfaces tend to minimize writing code). However, they can be extremely valuable and could be suitable in more specialized courses. Our framework for SCL applies to these specialized tools as well. Data management practices, such as specific organizational schemes or file size management, applies to specialized tools when you need to consider the filetype and memory allotment needed to utilize the tool properly. Furthermore, specialized tools can be used to help students familiarize themselves with the commonly used visualizations of the field in a simple and centralized way. Summarizing these practices for new students can expedite mentoring students in not only the use of these tools but the development of cognitive and material literacies as well.

Limitations

This study has a few potential limitations. The first of which is that interview participants were drawn from academic faculty and therefore do not reflect the practices of non-academic professionals. Previous work suggests that professionals in industry, particularly in engineering, are often exposed to a plethora of specialized tools (Rajlich, 2013). The relative abundance of general and specialized tool users in our study may not reflect how tools are used in industry, and therefore may not capture some of the norms professionals have for using specialized tools. Additionally, non-academic professionals are likely to have some different motivations than academic faculty. If this is the case there could be a unique set of SCL goals professionals have, as well as a subset of the SCL goals we found that do not apply in the same way outside of academia. Additionally, there is a limited number of participants in each academic discipline in this study. More participants would be needed to determine if there were more SCL goals that our participants did not mention. Another aspect that is not fully captured by our framework is the personal goals of individual lab members who are contributing to a group project. Our activity systems are defined by shared research goals, often set by the principal investigator and taken up by the research team. Individual members of the research team could have their own goals. For instance, a junior member of a research group could simultaneously focus on high-quality documentation and figure creation for research productivity, which is identified in this study, but could also use those same artifacts to persuade senior members of the research group of their skills and abilities. In this way, one practice could involve both productivity and persuasion but to meet different goals (research vs. personal).

These caveats notwithstanding, our work provides a framework to understand how researchers employ the social aspects of computation. Our framework forefronts STEM researchers' motivation and allows us to contextualize educational activities that address CT and authentic scientific practices.

Acknowledgements We would like to thank Ting Zhang for her feedback about the analysis of this data.

Author Contribution M. V., K. P., M. M., K. D., and A. C. B. conducted interviews and designed the interview protocol. C. C. analyzed data with feedback from M. F., T. E. W., and B. Z. C. C. wrote the paper with feedback and comments from M. F., T. E. W., and B. Z., T. E. W., and B. Z. conceived and supervised the project.

Funding This work was supported by the National Science Foundation (Grant numbers DGE-2222337 and DUE-2149957).

Data Availability Not applicable.

Declarations

Ethics Approval The conduct of this research was approved by the Rochester Institute of Technology Institutional Review Board HSRO #02042222.

Consent to Participate Informed consent was obtained from all participants included in this study.

Consent for Publication Not applicable.

Conflict of interest The authors declare no competing interests.

References

AAPT UCTF. (2016). *Recommendations for computational physics in the undergraduate physics curriculum* [Computational physics report]. https://www.aapt.org/Resources/upload/AAPT_UCTF_CompPhysReport_final_B.pdf

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *Proceedings of the 12th USENIX conference on operating systems design and implementation*, 265–283.

ABET. (2024). *Accreditation criteria & supporting documents.* https://www.abet.org/accreditation/accreditation-criteria/

Aizenman, J., King, C., Kling, T., Kober, G., Ramsey, L., Solomon, J., Waratuke, S., & Womack, C. (2024). A Liberal Arts curriculum that situates science while promoting STEM graduation. *Science & Education*. https://doi.org/10.1007/s11191-024-00532-0

American Association for the Advancement of Science. (1994). *Benchmarks for science literacy*. Oxford University Press.

Beheshti, E., Weintrop, D., Swanson, H., Orton, K., Horn, M., Jona, K., Trouille, L., & Wilensky, U. (2017). Computational thinking in practice: How STEM professionals use CT in their work. *American Education Research Association Annual Meeting 2017*.

Bialek, W., & Botstein, D. (2004). Introductory science and mathematics education for 21st-century biologists. *Science, 303*(5659), 788–790. https://doi.org/10.1126/science.1095480

Carnevale, A. P., Smith, N., & Melton, M. (2011). STEM: Science technology engineering mathematics. In *Georgetown University Center on Education and the Workforce*. Georgetown University Center on Education and the Workforce. https://eric.ed.gov/?id=ED525297

Charles, T., & Gwilliam, C. (2023). The effect of automated error message feedback on undergraduate physics students learning python: Reducing anxiety and building confidence. *Journal for STEM Education Research, 6*(2), 326–357. https://doi.org/10.1007/s41979-022-00084-4

Chen, C., Chen, H., Zhang, Y., Thomas, H. R., Frank, M. H., He, Y., & Xia, R. (2020). TBtools: An integrative toolkit developed for interactive analyses of big biological data. *Molecular Plant, 13*(8), 1194–1202. https://doi.org/10.1016/j.molp.2020.06.009

DeJarnette, A. F., Larrison, C., Rollmann, S. M., Vanderelst, D., Layne, J. E., & Hutchinson, A. E. (2022). Students' use of computational thinking practices in an undergraduate biology-engineering course. *Journal for STEM Education Research, 5*(1), 53–77. https://doi.org/10.1007/s41979-021-00058-y

diSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. MIT Press.

Engeström, Y. (2000). Activity theory as a framework for analyzing and redesigning work. *Ergonomics, 43*(7), 960–974. https://doi.org/10.1080/001401300409143

Engeström, Y. (1987). *Learning by expanding: An activity-theoretical approach to developmental research*. Orienta-konsultit.

Finley, A. P. (2023). *The career-ready graduate: What employers say about the difference college makes*. American Association of Colleges and Universities. https://eric.ed.gov/?id=ED634818

Gambrell, J., & Brewe, E. (2024). Analyzing interviews on computational thinking for introductory physics students: Toward a generalized assessment. *Physical Review Physics Education Research, 20*(1), 010128. https://doi.org/10.1103/PhysRevPhysEducRes.20.010128

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Hurt, T., Greenwald, E., Allan, S., Cannady, M. A., Krakowski, A., Brodsky, L., Collins, M. A., Montgomery, R., & Dorph, R. (2023). The computational thinking for science (CT-S) framework: Operationalizing CT-S for K–12 science education researchers and educators. *International Journal of STEM Education, 10*(1), 1. https://doi.org/10.1186/s40594-022-00391-7

ISTE & CSTA. (2011). *Operational definition of computational thinking for K-12 education*. https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Juavinett, A. (2020). Learning how to code while analyzing an open access electrophysiology dataset. *Journal of Undergraduate Neuroscience Education, 19*(1), A94–A104.

Juavinett, A. L. (2022). The next generation of neuroscientists needs to learn how to code, and we need new ways to teach them. *Neuron, 110*(4), 576–578. https://doi.org/10.1016/j.neuron.2021.12.001

Juškevičienė, A., & Dagienė, V. (2018). Computational thinking relationship with digital competence. *Informatics in Education - an International Journal, 17*(2), 265–284.

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). *A framework for computational thinking based on a systematic research review*. https://www.semanticscholar.org/paper/A-Framework-for-Computational-Thinking-Based-on-a-Kalelio%C4%9Flu-G%C3%BClbahar/e4f94b4e16a87e4b815893388c63c096038a69b8

Kilamo, T., Hammouda, I., & Chatti, M. A. (2012). Teaching collaborative software development: A case study. *2012 34th International Conference on Software Engineering (ICSE)*, 1165–1174. https://doi.org/10.1109/ICSE.2012.6227026

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020a). Computational thinking is more about thinking than computing. *Journal for STEM Education Research, 3*(1), 1–18. https://doi.org/10.1007/s41979-020-00030-2

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020b). On computational thinking and STEM education. *Journal for STEM Education Research, 3*(2), 147–166. https://doi.org/10.1007/s41979-020-00044-w

List, M., Ebert, P., & Albrecht, F. (2017). Ten simple rules for developing usable software in computational biology. *PLOS Computational Biology, 13*(1), e1005265. https://doi.org/10.1371/journal.pcbi.1005265

Lockwood, E., & Mørken, K. (2021). A call for research that explores relationships between computing and mathematical thinking and activity in RUME. *International Journal of Research in Undergraduate Mathematics Education, 7*(3), 404–416. https://doi.org/10.1007/s40753-020-00129-2

Lodi, M. (2020). Informatical thinking. *Olympiads in informatics*, 113–132. https://doi.org/10.15388/ioi.2020.09

Mason, S. (2020). Collaborative learning in computing education: Faculty perspectives and practices. *Proceedings of the 2020 ACM conference on international computing education research*, 136–146. https://doi.org/10.1145/3372782.3406254

Moskovitz, C., & Kellogg, D. (2011). Inquiry-based writing in the laboratory course. *Science, 332*(6032), 919–920. https://doi.org/10.1126/science.1200353

Nardi, B. A. (Ed.). (1996). *Context and consciousness: Activity theory and human-computer interaction*. MIT Press.

NGSS Lead States. (2013). *Next generation science standards: For states, by states*. National Academies Press. http://www.nextgenscience.org/

Odden, T. O. B., & Burk, J. (2020). Computational essays in the Physics classroom. *The Physics Teacher, 58*(4), 252–255. https://doi.org/10.1119/1.5145471

Odden, T. O. B., & Malthe-Sørenssen, A. (2021). Using computational essays to scaffold professional physics practice. *European Journal of Physics, 42*(1), 015701. https://doi.org/10.1088/1361-6404/abb8b7

Odden, T. O. B., Lockwood, E., & Caballero, M. D. (2019). Physics computational literacy: An exploratory case study using computational essays. *Physical Review Physics Education Research, 15*(2), 020152. https://doi.org/10.1103/PhysRevPhysEducRes.15.020152

Palmer, D. C. (2015). Visualization and analysis of crystal structures using CrystalMaker software. *Zeitschrift Für Kristallographie - Crystalline Materials, 230*(9–10), 559–572. https://doi.org/10.1515/zkri-2015-1869

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., da Leprevost, F., Fufezan, V. C., Ternent, T., Eglen, S. J., Katz, D. S., Pollard, T. J., Konovalov, A., Flight, R. M., Blin, K., & Vizcaíno, J. A. (2016). Ten simple rules for taking advantage of git and GitHub. *PLOS Computational Biology, 12*(7), e1004947. https://doi.org/10.1371/journal.pcbi.1004947

Pevzner, P., & Shamir, R. (2009). Computing has changed biology—Biology education must catch up. *Science, 325*(5940), 541–542. https://doi.org/10.1126/science.1173876

Rajlich, V. (2013). Teaching developer skills in the first software engineering course. *2013 35th International conference on software engineering (ICSE)*, 1109–1116. https://doi.org/10.1109/ICSE.2013.6606661

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLOS Computational Biology, 9*(10), e1003285. https://doi.org/10.1371/journal.pcbi.1003285

Skuse, B. (2019). The third pillar. *Physics World, 32*(3), 40. https://doi.org/10.1088/2058-7058/32/3/33

Taschuk, M., & Wilson, G. (2017). Ten simple rules for making research software more robust. *PLOS Computational Biology, 13*(4), e1005412. https://doi.org/10.1371/journal.pcbi.1005412

Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., in 't Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C., & Plimpton, S. J. (2022). LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, *271*, 108171. https://doi.org/10.1016/j.cpc.2021.108171

Vee, A. (2017). *Coding literacy: How computer programming is changing writing*. MIT Press.

Vygotsky, L. S., & Cole, M. (1978). *Mind in society: Development of higher psychological processes*. Harvard University Press.

Wang, C., Shen, J., & Chao, J. (2022). Integrating computational thinking in STEM education: A literature review. *International Journal of Science and Mathematics Education, 20*(8), 1949–1972. https://doi.org/10.1007/s10763-021-10227-5

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., Haddock, S. H. D., Huff, K. D., Mitchell, I. M., Plumbley, M. D., Waugh, B., White, E. P., & Wilson, P. (2014). Best

practices for scientific computing. *PLOS Biology, 12*(1), e1001745. https://doi.org/10.1371/journal.pbio.1001745

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology, 13*(6), e1005510. https://doi.org/10.1371/journal.pcbi.1005510

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society a: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Yamagata-Lynch, L. C. (2010). *Activity systems analysis methods: Understanding complex learning environments*. Springer.

Zagalsky, A., Feliciano, J., Storey, M.-A., Zhao, Y., & Wang, W. (2015). The emergence of GitHub as a collaborative platform for education. *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, 1906–1917. https://doi.org/10.1145/2675133.2675284

Zohrabi Alaee, D., & Zwickl, B. M. (2023). Challenges and outcomes in remote undergraduate research programs during the COVID-19 pandemic. *Physical Review Physics Education Research, 19*(1), 010135. https://doi.org/10.1103/PhysRevPhysEducRes.19.010135

Zwickl, B. M., Ikoro, V., & Allie, S. (2023). Characterizing lab environments using activity theory. In *The International handbook of physics education research: Teaching physics* (pp. 10–1–10–30). AIP Publishing LLC. https://doi.org/10.1063/9780735425712_010

Authors and Affiliations

Christian Cammarota[1,2] · Michael Foster[1,3] · Mike Verostek[2,4] · Kayleigh Patterson[2] · Mikayla MacIntyre[3] · Kimberly Dorsey[5] · Andrea Camacho-Betancourt[6] · Tony E. Wong[3] · Benjamin Zwickl[2,7]

✉ Christian Cammarota
christian.cammarota@gmail.com

[1] Thomas H. Gosnell School of Life Sciences, Rochester Institute of Technology, Rochester, NY, USA

[2] School of Physics and Astronomy, Rochester Institute of Technology, Rochester, NY, USA

[3] School of Mathematics and Statistics, Rochester Institute of Technology, Rochester, NY, USA

[4] Department of Physics and Astronomy, University of Rochester, Rochester, NY, USA

[5] Computational and Applied Mathematics, Rice University, Houston, TX, USA

[6] Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

[7] Center for Computing in Science Education, University of Oslo, Oslo, Norway