Magana *et al.*
International Journal of STEM Education (2024) 11:34
https://doi.org/10.1186/s40594-024-00494-3

International Journal of STEM Education

RESEARCH

Open Access

Scaffolded team-based computational modeling and simulation projects for promoting representational competence and regulatory skills

Alejandra J. Magana[1*], Joreen Arigye[1], Abasiafak Udosen[1], Joseph A. Lyon[1], Parth Joshi[1] and Elsje Pienaar[1]

Abstract

Background This study posits that scaffolded team-based computational modeling and simulation projects can support model-based learning that can result in evidence of representational competence and regulatory skills. The study involved 116 students from a second-year thermodynamics undergraduate course organized into 24 teams, who worked on three two-week-long team-based computational modeling and simulation projects and reflected upon their experience.

Results Results characterized different levels of engagement with computational model-based learning in the form of problem formulation and model planning, implementation and use of the computational model, evaluation, and interpretation of the outputs of the model, as well as reflection on the process. Results report on students' levels of representational competence as related to the computational model, meaning-making of the underlying code of the computational model, graphical representations generated by the model, and explanations and interpretations of the output representations. Results also described regulatory skills as challenges and strategies related to programming skills, challenges and strategies related to meaning-making skills for understanding and connecting the science to the code and the results, and challenges and strategies related to process management mainly focused on project management skills.

Conclusion Characterizing dimensions of computational model-based reasoning provides insights that showcase students' learning, benefits, and challenges when engaging in team-based computational modeling and simulation projects. This study also contributes to evidence-based scaffolding strategies that can support undergraduate students' engagement in the context of computational modeling and simulation.

Keywords Representational competence, Computational modeling, Model-based learning, Regulatory skills, Scaffolding, Team-based projects, Undergraduate STEM education

Introduction

Model-based reasoning is central for scientists, engineers, and mathematicians to draw upon foundational scientific or mathematic principles to model, simulate, and understand systems (Aurigemma et al., 2013; Lehrer & Schauble, 2006). Model-based reasoning involves cognitive processes, such as analogical reasoning, visual imagery, representation, and thought experiments, with the goal of constructing new conceptual structures

*Correspondence:
Alejandra J. Magana
admagana@purdue.edu
[1] Purdue University, West Lafayette, IN, USA

(Nersessian, 2002a). Such conceptual structures are often represented as models used to understand or explain an ill-known phenomenon or system (Ifenthaler & Seel, 2013). Thus, to promote the effective use of models in education, instructors must apply model-based learning, which involves teaching strategies that engage students in model-based reasoning processes (Clement & Rea-Ramirez, 2008). Such model-based learning aims to engage learners in co-construction processes building on instructor-generated and student-generated model elements (Hallström & Schönborn, 2019; Hallström et al., 2023). Models, once expressed as external representations, can assist students in making sense of phenomena or systems (Develaki, 2017; Johnson-Laird, 1995). External representations can take the form of diagrams, equations, and computer simulations (Magana et al., 2020; Nersessian, 2009). Learning the uses of these representations is referred to as acquiring representational competence (Stull et al., 2016). Thus, to develop model-based learning, students must be (a) exposed to the process of creating, testing, revising, and using models that reflect their understanding of phenomena or systems (Malone et al., 2018; Schwarz & White, 2005), (b) supported so that they develop representational competence as an ability to understand, create, and flexibly utilize multiple types of models (Avargil & Piorko, 2022), (c) engaged in metacognitive processes for monitoring and evaluating progress (Cox, 2005; Lyon & Magana, 2021), and (d) guided through regulatory processes while working with others (DiDonato, 2013; Garrison & Akyol, 2015).

Students pursuing majors in STEM are expected to engage in a specific form of model-based learning—computational model-based learning—where, in addition, modeling and simulation skills are applied to various problem-solving contexts across disciplinary curriculums in foundational and advanced courses (Nersessian & Patton, 2009; Sun et al., 2006). Empirical research has identified that students benefit from their conceptual learning when engaging in computational model-based learning (Malone et al., 2018). However, since the 80s, education research has identified over and over the complex nature of learning computational modeling and simulation skills (e.g., Cheah, 2020; Mayer, 1985). In recent years, discipline-based education researchers have begun to delineate critical points and key challenges in the development of students' computational skills to support model-based learning processes in the context of modeling and simulation (Magana et al., 2020; Turner et al., 2011).

Specific challenges in undergraduate STEM education are centered on developing computational model-based learning (Magana et al., 2017, 2020). This is attributed in part to students needing to integrate programming skills with mathematical knowledge and disciplinary concepts (Magana et al., 2013, 2017). Specifically, prior research indicates that in addition to programming challenges, certain practices, particularly those that demand the integration of differing representations, i.e., physical, mathematical, and algorithmic, such as those that occur in modeling and simulation, run significant risks of causing "cognitive overload" (Magana et al., 2013; Vieira et al., 2016b). Even advanced postgraduate engineers in research settings face significant difficulty in the abstraction and transfer of knowledge between physical and algorithmic representations (Madamanchi et al., 2018). A significant limitation in our evolving understanding of the progression of undergraduate students in computational and model-based skills is that, to date, most studies have been limited to students in K-12 settings (Hurt et al., 2023). Fortunately, work in engineering and physics education has started to document effective ways for delivering computational model-based instruction through scaffolded pedagogy (e.g., Fennell et al., 2020; Lyon & Magana, 2021; Medeiros et al., 2018; Odden et al., 2019).

As part of our educational initiatives in the context of higher education, we are focusing on the need to integrate computational model-based learning sooner and more often in the undergraduate STEM curriculum. Specifically in the context of engineering majors, it is important for students to develop computational model-based reasoning because bodies of accreditation, such as ABET and the Washington and Dublin Accords, have consistently identified as an outcome for students to possess an ability to use the computational techniques, skills, and modern engineering tools necessary for engineering practice. However, research has identified that the integration of computational model-based learning is more common at the graduate level, with fewer opportunities for undergraduate students to practice those skills (Aurigemma et al., 2013; Magana & Mathur, 2012). More recently, educators and discipline-based education researchers in physics education (e.g., Mashood et al., 2022), biology education (e.g., Helikar et al., 2015), and engineering education (e.g., Ortega-Alvarez et al., 2018), among others, have documented the opportunities and challenges of integrating computational model-based learning in the context of at the undergraduate level. Research suggests that the integration of certain forms of scaffolding (i.e., strategies that can help learners succeed in solving problems that are otherwise too difficult for them to achieve) could effectively support students in successfully completing their projects (Quintana et al., 2004).

Specifically, in this study, we characterized and evaluated undergraduate students' evidence of computational model-based learning while completing scaffolded computational modeling and simulation projects working in

teams. In addition, we identified how that engagement resulted in (a) representational competence in terms of the code representing the computational model and the output of the code and (b) regulatory processes students applied to overcome learning challenges and working with others as they approached the project solution as a team. This study posited that scaffolded computational modeling and simulation projects could facilitate computational model-based learning, resulting in representational competence in the form of mappings between models and regulatory skills to overcome learning and collaboration challenges. To test this conjecture, the research question for this study is: *How do scaffolded team-based computational modeling and simulation projects engage students in model-based learning, resulting in representational competence and regulatory skills?*

Background literature

In an inquiry context, learning includes engaging in processes of creating models embodying aspects of theory and data, evaluating models using criteria, such as accuracy and consistency, revising models to accommodate new theoretical ideas or empirical findings, and engaging in discussions about models and the process of modeling (Schwarz & White, 2005). While often the models created take the form of drawings (e.g., Quillin & Thomas, 2015), models can also take the form of mathematical formulations (Lehrer & Schauble, 2003), or computer models (e.g., Araujo et al., 2008). When combined with computer modeling, inquiry learning can be enhanced by providing tools to represent ideas formally, configuring and running a model to compute the values of the variables as they develop over time, interpreting the outputs of models by noticing the course of events predicted by the model, evaluating the calculated values by comparison with predictions or data collected in an experiment, and iteratively revising the model to improve it based on data or new knowledge (Borkulo et al., 2008; Magana et al., 2020).

In the context of STEM education, inquiry learning is often complemented with problem-solving processes. In such instances, modeling is accompanied by simulation processes to iteratively revise systems. That is, while modeling involves producing a representation of the inner workings of a phenomenon or system, simulation involves operating the model by exploring and reconfiguring it iteratively in cyclic steps that allow individuals to make predictions about and alter the systems under study (Shiflet & Shiflet, 2014). However, in addition to inquiry and problem-solving processes, engaging in modeling and simulation may also require the application of computational skills (Mashood et al., 2022) and mathematical skills (Penner, 2000).

Although the benefits of integrating computational modeling and simulation processes at the undergraduate level can result in meaningful forms of learning (Chao, 2016), the integration of multiple knowledge and skills also involves learning difficulties (Magana et al., 2013; Vieira et al., 2017). Such difficulties include: (1) the level of interaction with real-world systems makes these mathematical models inflexible or fragile, (2) models based on calculus may be beyond the mathematical capabilities of learners, and (3) highly abstract mathematical models are far from the student's everyday knowledge of the world (Penner, 2000). Thus, new pedagogical and scaffolding strategies are needed to support the effective integration of computational modeling and simulation, resulting in effective learning and engagement individually and with others.

Computational model-based learning

Model-based reasoning refers to the practice of engaging in analogical reasoning, visual imagery, representation, and thought experimenting (Nersessian, 2002a), with the goal of making sense of phenomena or systems through different forms of external representations (Johnson-Laird, 1995; Zwickl et al., 2015). To promote model-based reasoning, educators must engage learners in social processes of model-based learning involving creating, testing, revising, and using externalized models (e.g., diagrams, graphs, drawings, mathematical expressions, and computational simulations) (Nersessian, 2008; Zwickl et al., 2015). The acquired ability of students to understand, develop, and flexibly utilize multiple types of representation models is referred to as representational competence (Avargil & Piorko, 2022; Stull & Hegarty, 2016). In this context, representational competence involves learners' ability to (a) abstract physical phenomena into some form of representational model alluding to a target system or phenomenon, (b) identify assumptions, simplifications, and limitations, (c) connect those models to known principles, concepts, or data, and (d) adapt them effectively during problem-solving episodes (Nersessian, 2002b; Zwickl et al., 2015). As learners engage in model-based learning, they apply and, at the same time, acquire or refine their existing knowledge (Edelsbrunner et al., 2023) and representational competence (Magana et al., 2020). Representational competence involves learners' ability to translate among multiple representations and delineate conceptual mappings between them (Stieff et al., 2016). Specifically, for analyzing a problem or phenomenon and formulating a model, individuals need to identify the variables of a model and the relationship while creating an initial representational

model. It is also necessary to determine variables and relationships between variables and determine equations and function interpretations of those variables, where prior knowledge relevant to the phenomenon to be modeled is critical (Löhner et al., 2005). To determine the model's behavior, physical laws are determined by the underlying equations (Sins et al., 2005). Such equations and functions need to be mapped into an algorithm and implemented in the model using different methods, techniques, and computational tools (Shiflet & Shiflet, 2014). Equations are solved numerically and programed those into functions embedded within an executable simulation (Magana et al., 2017). Validation and verification processes also require inspection of representations to determine whether the model is consistent with data obtained from experiments or with theoretical descriptions of the expected behavior (Sins et al., 2005). Finally, once the model is evaluated, individuals must execute the model and interpret the outputs of the model. In this process, prior knowledge influences the quality of the models students construct (Sins et al., 2005), and new knowledge is achieved through the processes of creating, connecting, and refining representations (Nersessian, 2002b).

However, in the context of computational modeling and simulation activities, research has documented the challenges students may experience when engaged in computational model-based learning. Specifically, students experience challenges in identifying mathematical formulations, as those are commonly based on differential or partial differential equations (Penner, 2000). Students also experience challenges in mapping mathematical equations into algorithmic representations and algorithmic representations into some form of code representing a computational model (Magana et al., 2017). These challenges have not only been attributed to insufficient prior conceptual understanding (Zwickl et al., 2015), but also to limited programming skills (Vieira et al., 2016a, 2016b). Students may experience difficulties regarding algorithmic logic, the use of multiple computational platforms to program, and the lack of non-technical skills, such as group work (Santos et al., 2020).

To overcome challenges during model-based learning, students must apply regulatory skills (Munahefi & Waluya, 2018). One type of regulatory skill is self-monitoring, which involves students assessing their learning and learning processes (Hawe et al., 2019). This regulatory skill is an important initial stage for students to evaluate their progress and make necessary changes to guide their course of learning more effectively (Zimmerman & Paulsen, 1995). Another type of regulatory skill needed during model-based learning is cognition regulation (Tee et al., 2021). Specifically, cognition regulation involves one's proactive selection and use of various cognitive strategies, such as critical thinking skills and elaboration strategies, to support reasoning abilities (Tee et al., 2021). While critical thinking skills involve the use of strategies to devise alternative solutions, questions, reflection, and mathematical thinking, elaboration skills involve the use of strategies to extract and summarize main ideas, connecting, relating, and applying knowledge by pulling together the acquired information from different resources to facilitate reasoning (Tee et al., 2021).

Thus, evidence-based strategies are needed to help learners engage in conceptual mappings informed by prior knowledge as well as regulatory skills that help them identify alternative solutions and connect relevant ideas. Evidence-based strategies must engage learners in (a) implementing iterative modeling cycles promoting conceptual knowledge and disciplinary practices (Rittle-Johnson et al., 2001), (b) building models that make possible the re-representation, manipulation, and refinement of knowledge (Chandrasekharan & Nersessian, 2015), (c) making explicit connections between knowledge and modeling experiences to create arguments in the form of explanations of why and how (Zhang et al., 2006), (d) probing the model behavior to identify causal relations by making some attributes of the phenomenon or system more salient than others (Chandrasekharan et al., 2012; Izsak, 2004), and (e) facilitating collaboration and regulatory processes to engage learners in the social processes involved in model-based reasoning (Liu et al., 2018; Sabourin et al., 2013; Şen et al., 2015).

Evidence-based strategies for teaching computational model-based learning.

It is clear that learners need guidance informed by evidence-based strategies that support them in socially engaging in computational model-based learning. By doing so, learners can make explicit connections between multiple forms of representations needed during the computational modeling and simulation processes, as well as the application and acquisition of representational competence and regulatory skills. Computational model-based learning can be best delivered through modeling and simulation practices (Magana, 2024). In doing so, it is important to consider (a) delivering these practices within the context of disciplinary courses so that students apply and at the same time reinforce scientific or engineering knowledge, (b) sequencing the activities from simple to complex, and practice skills in a variety of situations, and (c) implementing elements of the sociology of learning (Magana, 2024). Regarding the sociology of learning, it is important for computational model-based learning to occur in the context of realistic tasks and in accordance with communities of practice. Thus, it is important to engage students in collaborative and

cooperative forms of learning that closely resemble professional practice and that can also support the development of teamwork skills (Collins et al., 2018).

Research on learning computational modeling and simulation also suggests that students need support outside of the lecture in the form of scaffolding (Magana, 2024). Scaffolding refers to evidence-based strategies that can help learners succeed in solving problems that are otherwise too difficult for them to achieve (Quintana et al., 2004). Scaffolding is based on the zone of proximal development, which refers to the idea that learning is most effective when learners actively engage in tasks they can do with some guidance (Vygotsky & Cole, 1978). There may be tasks that learners can perform with no help and tasks that they simply cannot do. Learning activities should be designed so learners take advantage of the tasks that they can complete on their own but are moved toward activities or tasks they cannot complete by providing guidance in the form of scaffolding (Quintana et al., 2004). By presenting scaffolded learning challenges and then gradually decreasing those, learners can acquire new knowledge and skills as learners become more capable (Wood et al., 1976). Scaffolding also promotes motivation, as it can make the solution to the posed problem achievable for the students (Deci et al., 2017). There are different forms of scaffolding that differ in their (a) purpose, such as for supporting procedural, conceptual, metacognitive, or strategic processes; (b) form facilitated interaction, such as static or fixed guidelines or dynamic interaction methods as a response to learners' actions; and (c) sources such as peers, teachers, or technology (Kim & Hannafin, 2011). These could take the form of individual activities, such as exercises, worked-out examples, live coding, visualizations, or prompts (Luxton-Reilly et al., 2018; Sanchez-Pena et al., 2022). Scaffolds can have multiple functions, including reducing the degrees of freedom of a task, demonstrating how to approach a task, showing directions, and making critical features salient (Kim & Hannafin, 2011). Thus, scaffolding is needed to support computation practices in science and engineering among novice learners, particularly those that effectively engage students (Magana et al., 2019; Psycharis et al., 2014).

Recent empirical studies in engineering and science education have started to characterize specific scaffolding in assisting student learning in the context of computational model-based learning (Magana, 2024). Specifically, the scaffolding includes sense-making, articulation, process management, and reflection, as proposed by Quintana et al. (2004). Specifically, (1) sense-making scaffolding supports ways to understand real-world phenomena or experiences and transform those into formal representations; (2) articulation scaffolding supports ways of making thinking explicit, synthesizing explanations, and creating arguments; (3) process management scaffolding supports the use of disciplinary practices, problem-solving, and strategic approaches, and (4) reflection scaffolding, supports the reviewing, reflecting on, and evaluating results, including elements of collaboration.

For instance, sense-making scaffolding, in the form of worked-out examples consisting of step-by-step sample solutions, has been identified as highly effective. In addition, articulation scaffolding in the form of in-code comments as an explanation strategy has been identified as an effective approach to engaging students with the worked-out examples (Vieira et al., 2017). Project templates have also been identified as useful process management scaffolds to guide students through the problem-solving processes, such as the modeling and simulation cycle (Lyon et al., 2022; Vieira et al., 2016b). When guided through processes using project templates, students can structure their work. For instance, students can be asked to express their understanding of the problem explicitly during the initial problem recognition phase. During this phase, the students can be guided to create a plan to work toward a solution. The student uses verbal and mathematical representations for this purpose. During the problem-framing phase, the students create a solution in the form of an algorithm instantiated as a program. In the problem synthesis phase, the student completes the plan by evaluating the solution according to both instructor-provided and student-generated criteria. The implementation of test cases is one of the most challenging phases for students and one of the most valuable for those who go on to programming practice in any context (Vieira et al., 2015).

Finally, to engage in individual learning and collaborative problem-solving, students need to also engage in metacognitive and regulatory processes, so they use elaboration and learning strategies and apply critical thinking skills (Tee et al., 2021). Effectively managing model-based reasoning requires extra metacognitive processes to regulate the complexity of modeling tasks while also working on them with others (Oyelere et al., 2021). Thus, an argument has been made that supporting regulatory, co-regulatory, or metacognitive processes is also needed for supporting computational model-based learning while working individually and with others (Magana et al., 2019), where reflection scaffolding, in the form of reflective practices, is one of the primary mechanisms for promoting these processes (Arigye et al., 2024; Jaiswal et al., 2021; Peters & Kitsantas, 2010).

Methods

This study used a multimethod approach to answer our guiding research question: *How do scaffolded team-based computational modeling and simulation projects engage students in model-based learning, resulting in representational competence and regulatory skills?* A multimethod approach was deemed adequate for the study as it allowed us to flexibly characterize and interrelate evidence of model-based learning, representational competence, and regulatory skills, as promoted through our learning design. Multimethod studies also support triangulation by incorporating multiple data sources and analytical approaches (Vivek & Nanthagopan, 2021).

Context and participants

The context of this study was a second-year thermodynamics course titled Thermodynamics in Biological Systems II offered at a large Land Grant Public Midwestern University. This course was deemed adequate for conducting this investigation for various reasons. First, thermodynamics is a course required for most engineering students whose topics have been characterized as difficult to understand (Abdullah et al., 2013); second, the integration of computational methods has been identified as a feasible pedagogical approach for teaching thermodynamics (Vieira et al., 2018); and third, there is a need to expose undergraduate students, especially in their second and third year of their undergraduate majors, to computational methods and practices, such as modeling and simulation (Magana & Mathur, 2012). In their first year of their undergraduate major, engineering students are introduced to programming concepts, while in their last year, they apply those skills in the context of their capstone courses. However, there are fewer opportunities for students to practice computational modeling and simulation practices in the middle years and in the context of their disciplines.

The course is required for students majoring in biomedical, agricultural, and biological engineering. The major objective of the course is to understand and exploit basic principles of thermodynamics as they apply to biological systems and biological processes. Prerequisites of the course include an introductory thermodynamics course where students learned about (a) the fundamentals of steady-state mass and energy balances for reacting and non-reacting processes, including multiple unit operations emphasizing living systems and bioprocessing, (b) the applications of the first law conservation of energy to biological systems, energy conversion systems, and the environmental impacts of energy production and (c) the development of engineering problem-solving skills via MathCad and MatLab software. A second prerequisite includes a mathematics course where students learn about multivariate and vector calculus. Also, in their first year, students were required to take an introductory course where they learned and applied basic programming concepts and constructs via the MATLAB programming language. This introductory course also introduced best practices for teamwork skills, and students completed a team-based project.

The objectives of the thermodynamics course are for students to understand basic principles of mass, energy, and entropy balance equations; apply these basic principles to solve problems in biological and biomedical engineering; gain knowledge of the main factors that determine numerical values of physical properties associated with bioprocesses; and solve biological and biomedical problems for thermodynamic processes using computer code. This is a required course for all majors in biomedical and biological engineering. Students typically enroll in this course during the second year of their undergraduate studies. Exceptions occur when students participate in voluntary or extracurricular programs such as co-ops or long-term internships. These students have been exposed to related foundational courses, such as (a) a first-year engineering course where they learn about the problem-solving process and the use of scientific computation in this context, (b) mathematics courses where they learn linear algebra and differential equations that are used within this course, and (c) in the case of biological engineering students, an introductory thermodynamics course where they are introduced to the basic thermodynamic principles.

The participants were 116 undergraduate students pursuing majors in biomedical, agricultural, and biological engineering. According to institutional data, in 2021–2022, 56% of the students were women, and 44% of the students were men. The majority of the students were White 67%, followed by Asian 21%, International 10%, more than two races 6%, Hispanic or Latino 3% and Black or African American 2%. Given this is a required course for the major plan of study, the participants were representative of the program as a whole. The students were organized into a total of 24 teams, each with four or five members. The students were randomly assigned to their teams, and the teams remained the same throughout the duration of the semester. Thus, the same team was consistent across all projects, and the three projects were submitted as a team.

Learning design

The learning design for this study aimed to promote computational model-based learning that would guide students to enact educational practices that support the development of representational competence and the

application of regulatory skills in the context of computational modeling and simulation of biological systems. For this, three two-week-long computational modeling and simulation projects were implemented throughout the semester. Specifics on how the scaffolding was delivered in the context of the computational modeling and simulation projects are presented in Table 1.

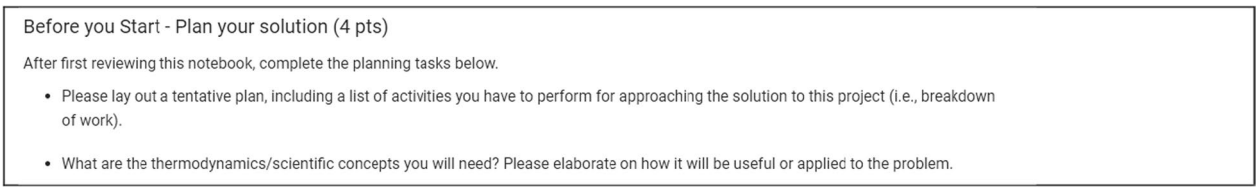
To deploy the computational modeling and simulation projects and, at the same time, embed the pedagogical supports described in Table 1, we used computational notebooks. The computational notebooks provided detailed explanations, guidance, and scaffolding throughout the project solution. Computational notebooks are web-based solutions that allow for authoring and executing code within a single document (Wang et al., 2019). Computational documents enable the delivery of programming content that is more accessible for teaching and learning by supporting the visualization of code execution, along with other forms of multimedia, including text, images, and videos (Granger & Pérez, 2021). Computational notebooks can also provide a learning advantage for embedding computational pedagogy (Sanchez-Pena et al., 2022).

The three projects were conceived by emphasizing the four educational practices supporting the development of model-based learning by exposing learners to the processes of formulating, implementing, using, and evaluating models. All three projects were scaffolded in a similar manner. The implementation of scaffolding is shown in Figs. 1, 2, 3, 4. Specifically, Fig. 1 provides an example of process management scaffolding aimed at guiding the students to define an initial plan in the form of a procedure and to recall the domain knowledge they would need to approach the implementation of their solutions. In Fig. 2, sense-making scaffolding was provided as guidance to the learners in the form of a template of code, providing some direction in the variables and functions needed to implement the solution. This form of scaffolding also elicited students to comment on the code to explain their thinking and decision-making processes.

Once the model was implemented, the next step for the students was to use the model to solve the biological problem. For this, students were prompted to explain their understanding of the outputs of the computational models in terms of the biological phenomenon. For this, Fig. 3 presents an example of articulation scaffolding that prompted students to explain the meaning of the graphs the models generated and to translate those observations into an explanation of the phenomenon in question and a solution to the biological problem. Finally, students were also provided with reflection scaffolding that had two purposes. One was for students to engage in a metacognitive reflection

Table 1 Specifics of sense-making, articulation, process management, and reflection scaffolding

Type	Description
Sense-making	Students were guided throughout the inquiry process one step at a time. The tasks were organized from simple to complex, and examples of outputs and test cases were provided throughout the process to evaluate their own implementations. Students were also guided in their implementation of their models by writing in-code comments
Articulation	Students were prompted to explain their thinking throughout the process. Specifically, students were asked to add in-code comments to explain their computational solutions. Students were also prompted to include explanations and arguments for each of the outputs of the graphical representations
Process management	Process management included an explicit planning stage for the project. Process management also consisted of support to facilitate teamwork coordination. Supports to facilitate teamwork coordination included signing team contracts at the beginning of the semester
Reflection	Students were prompted to reflect on and evaluate their results and processes individually and in teams. Individual reflections focused on lessons learned, experienced challenges, and ways to overcome challenges. Reflections at the team level focused on aspects of planning, monitoring, and evaluating processes


Before you Start - Plan your solution (4 pts)

After first reviewing this notebook, complete the planning tasks below.

- Please lay out a tentative plan, including a list of activities you have to perform for approaching the solution to this project (i.e., breakdown of work).
- What are the thermodynamics/scientific concepts you will need? Please elaborate on how it will be useful or applied to the problem.

Fig. 1 Sample of process management scaffolding consisting of prompts aimed at eliciting the analyzing the problem and planning and formulating a model

```
[ ]
    ## Hint: Remember to comment your code and use good coding practice.
    ##    If you write good comments it's easier for us to figure out what you're
    ##    trying to do and give partial credit if you make a mistake.
    ##    Example of bad comment: This is a for loop
    ##    Example of good comment: This is a for loop used to calculate...
    ##                             because of ...


    import numpy as np
    from scipy.integrate import odeint
    import matplotlib.pyplot as plt
    import math as m


    k_toxProd         = ??

    k_ToxDecay       = ??
    k_neut  = ??

    k_on  = ??
    k_off = ??

    k_antiDecay = ??
    k_E  =    ??

    beta_R = ??


    beta_1  = ??
    K1      = ??
    n1      = ??

    Toxin0            = ??
    ReceptorToxin0    = ??
    Receptor0         = ??
    Antitoxin0        = ??
```

```
y0  = [Toxin0, ReceptorToxin0, Receptor0, Antitoxin0]

z   = ??                          # ??
t   = np.linspace(0, 100.0, z)   # ??

#
def model(y, t):
  #
  T       = y[0]
  RT      = y[1]
  R       = y[2]
  A       = y[3]

  #
  dTdt  =

  dRTdt =

  dRdt  =

  dAdt =

  #
  return [dTdt, dRTdt, dRdt, dAdt]

#
soln  = odeint(model, y0, t)

#
Toxin             = soln[:, ??]
ReceptorToxin     = soln[:, ??]
Receptor          = soln[:, ??]
Antitoxin         = soln[:, ??]

#
MaxAllowableToxin = [??]*z

#
plt.figure(num=1)
plt.plot(t, Toxin,             label='Toxin')
plt.plot(t, ReceptorToxin,     label='RT')
plt.plot(t, Receptor,          label='Receptor')
plt.plot(t, Antitoxin,         label='Antitoxin')
plt.plot(t, MaxAllowableToxin, label='MaxToxin')

#
plt.xlabel('Time (s)')
plt.ylabel('Species Concentration (M)')
plt.title('System Timeseries')
```

Fig. 2 Sample of sense-making scaffolding consisting of templates of code and in-code commenting to guide learners through the process of model implementation

- For the plot that you generated in Task 2a with $k_{toxProd} = 0.5\frac{M}{s}$, please describe in detail what is physically happening from a biological standpoint, and why this gene module fails (or not) at this low toxin production rate.
- What did you learn about the characteristics of the gene module (i.e. the shape of the hill curve) that is not suitable for solving this problem?
- What would you change about the chosen gene so that they might work in future designs?

Fig. 3 Sample of articulation scaffolding consisting of prompts that elicited students' explanations of the outputs of the computational models

- Please describe the coding strategy you followed to solve this problem (specific loops, if/then statements, others)
- Please describe the biggest challenge you faced when implementing this coding strategy
- What skills did you have to develop to accomplish this project?

Fig. 4 Sample of reflection scaffolding consisting of prompts that elicited students' reflections on their learning processes, the challenges they experienced in their learning, and working in teams

of what they learned, identify the challenges students experienced in solving a particular project, along with the problem-solving strategies they implemented, and reflect on their team regulation processes by describing what worked, what didn't work, and what can be

improved in the next project or iteration. Examples of the reflection scaffolding are shown in Fig. 4.

Procedures and data collection methods

The specific first-time implementation of the scaffolding methods and the corresponding data collection occurred in the spring of 2022. The data collection methods consisted of student-generated team-based solutions to the scaffolded computational modeling and simulation projects that allowed us to characterize students' implementations of their model-based learning and the resulting outcomes in terms of representational competence and regulatory skills. Thus, instruments were defined so that each captured (a) students' products and processes for evaluating, creating, and communicating their models and outputs of their models, (b) students' representations and the interpretations of such representations, and (c) individual reflections and team-based retrospectives describing the application of their regulatory processes implemented to complete the projects.

Three team-based scaffolded computational modeling and simulation projects were implemented throughout the semester to fulfill one of the primary course objectives of applying basic principles of mass, energy, and entropy balance equations for solving biological and biomedical problems for thermodynamic processes using computer code. Table 2 presents specifics of the disciplinary and computational learning objectives for each of the projects. The titles of the projects were Could You Outrun a Dinosaur (P1), Toxin-Antitoxin System Design (P2), and Chemical Reactor Stability and Sensitivity (P3). The projects also increased in their level of difficulty in terms of the disciplinary and computational learning objectives. Each project was introduced following a similar instructional approach. Each project followed in-class lectures introducing the technical concepts required to complete the projects. When projects were assigned, a lecture was delivered that provided a description of (a) the context in which they were being asked to apply the technical concepts from lectures and (b) the full tasks they needed to complete within the project. Students had two weeks to complete the project outside of class. During that two-week period, students had an opportunity to attend TA and instructor office hours to receive project-specific help. The instructor provided opportunities during class times for project questions/clarifications based on patterns that the instructional team observed during office hours.

Project 1 was assigned on week two of the semester, Project 2 was assigned on week five of the semester, and Project 3 was implemented on week 11 of the semester. As shown in Table 2, the projects increased in their level of difficulty in terms of both disciplinary learning objectives and computational learning objectives. Since the projects were submitted as a team, there were 24 submissions per project, with a total of 72 projects analyzed.

For characterizing model-based learning, we qualitatively analyzed students' solutions to the projects involved observations of how students planned for their models, implemented and evaluated their models, used the models and interpreted the outputs of the models, and reflected on the process as shown in Figs. 1, 2, 3, 4, respectively. For characterizing representational competence, we evaluated students' alignments between (a) the implementation of the model and their explanations of the model in the form of in-code comments and (b) the

Table 2 Disciplinary and computational learning objectives for the course

Project	Disciplinary learning objectives	Computational learning objectives
P1: Could You Outrun a Dinosaur	• Collect and visualize data accounting for noise and uncertainty • Compute and interpret dimensionless quantities • Interpret and analyze data and resulting dimensionless quantities	• Enter and organize data • Plot/visualize data using arrays • Perform basic computation with arrays • Fit linear model to data
P2: Toxin-Antitoxin System Design	• Describe complex biological systems using models of genetic circuits • Characterize and describe dynamics in a given system of biological interactions • Evaluate and test possible system structures to achieve a stated goal	• Construct, solve, and visualize ordinary differential equations numerically using built-in tools • Implement "for loops" to iterate through arrays of parameters and perform computations • Plot/visualize data using arrays
P3: Chemical Reactor Stability and Sensitivity	• Construct and analyze mass and energy balances • Incorporate endo- and exothermic reactions into mass and energy balances • Interpret and characterize systems at and away from steady state • Predict operating conditions to achieve a stated goal in a bioreactor	• Construct, solve, and visualize ordinary differential equations numerically using built-in tools • Compute steady-state values of state variables using functions and built-in tools

Table 3 Individual and team-based reflection questions

Category	Questions
Individual reflection questions	• What skills did you have to develop to accomplish this project? • What aspects of this project were the most beneficial for your learning? • What challenges did you encounter in completing the project? • How did you overcome the challenges or remedy the problems encountered?
Team reflection questions	• What teamwork strategies that you chose worked? And why? • What teamwork strategies that you chose didn't work? And why? • What aspects do you think can be done better for the next project in terms of team coordination to address the issues that didn't work? (You must commit to having something to improve for the next project.)

outputs in the form of graphs generated by their models and their explanations of the outputs in terms of the biological system represented. To characterize the regulatory processes, we qualitatively analyzed and coded students' strategies to manage their learning challenges and teamwork processes as reported in their individual and team-based reflection questions listed in Table 3.

Data scoring and analysis methods

Computational model-based learning The data analysis procedure was initiated with a qualitative content analysis of the solutions that students submitted as a team, which consisted of the computational notebooks that embedded the scaffolding. The analysis started at the end of the semester, once all grades were submitted and all the dataset was compiled into teams. The projects were analyzed sequentially; that is, we first analyzed all 24 submissions corresponding to Project 1, then moved on to Project 2, and once completed, to Project 3. Each project was analyzed in full for a given team. For each project submission, the analysis started by chunking the data. To guide the process of chunking the data (Hsieh & Shannon, 2005), we used a directed content analysis approach where the definitions of model-based learning processes were used as guidance for the initial codes or, in this case, chunks of data (refer to Table 4). For this, we made a distinction between the portions of the submission corresponding to descriptions associated with the planning of the project, implementation of the model, interpretation of the outputs of the model, and the reflection processes that were submitted individually and also as a team, all within the same computational notebook.

We completed the chunking project for all 24 teams for Project 1. Once we identified the relevant portions of the project submissions aligned to how scaffolding supported students' model-based learning, we analyzed the samples qualitatively to explore similarities and differences between the samples of solutions with the goal of identifying patterns in the data. Following this process, we aimed to identify groups consisting of teams with similar outcomes resulting from computational model-based learning. We then selected representative extreme cases for further description in our findings with the goal of characterizing how students within teams demonstrated evidence of their modeling processes. For this, we present excerpts of students' solutions showcasing contrasting cases for Project 2. We opted to present samples from Project 2 in order to favor the readers' understanding of the interconnection of ideas within a single context. Specifically, we showcase how the scaffolding methods embedded within the computational notebooks elicited model-based learning processes that resulted in students' evidence of (a) analyzing the problem and planning and formulating the model, (b) implementing and evaluating the model to solve the problem, (c) using the model and interpreting the outputs generated after executing or

Table 4 Definitions of model-based learning processes that guided the chunking of the data

Model-based learning	Definition
Analyze the problem, plan, and formulate the model	Instances where students indicated recalling the disciplinary knowledge they would need to approach the project solution and consider an initial approach for solving the problem
Implement and evaluate the model to solve the problem	Instances where students indicated details of the model implementation, including equations, pseudocode, or code, such as variables, functions, and programming sequences, and the strategies they used for validating their models
Model use and interpretation of the model outputs	Instances where students explained the outputs of the models and the grounding and connections of those interpretations in the biological system or phenomenon
Report on the solution to the problem and reflect on the process	Instances where students reflected upon their learning individually and collectively

running the model, and (d) reflection processes reflecting regulatory strategies students implemented individually and as a team.

Representational competence Because of the identified similarities and differences in students' project submissions, we started to document those through a scoring rubric to capture and evaluate students' representational competence. For this, we paid particular attention to how students working in teams explained the connections between the models they generated and their understanding of such models. For this, an initial version of the rubric was created by two investigators who jointly analyzed the data. One investigator had extensive expertise in education research and assessment, and the second investigator, in addition, had disciplinary expertise in the domain of thermodynamics. The original version of the rubric was based on previous studies evaluating students' performance on their computational solutions to engineering problems (i.e., Shaikh et al., 2015; Vieira et al., 2016a, 2016b). As the scoring process started, the rubric was iteratively revised based on the characteristics of students' responses to the project. The rubric was flexible enough so all three projects could be scored with the same rubric. The revised and final version of the rubric is presented in Table 5.

Once the two investigators agreed that the rubric was representative of both the epistemic aims and the data itself, a subset of the assignments was reviewed by the two researchers and the rubric to evaluate the clarity of wording and the differentiation of the scoring categories. Once agreement (> 80%) was reached between the two researchers, all disagreements were resolved, and the disciplinary expert completed the scoring of the dataset. The interpretation for the scores assigned with the rubric presented in Table 4 was (a) between 1.0 and 2.0, the level of achievement in a particular skill was considered as below basic; (b) between 2.1 and 3.0, the level of achievement for a particular skill was considered as basic; (c) between 3.1 to 4.0, the level of achievement for a particular skill was considered as proficient; and (d) between 4.1 to 5.0 the level of achievement for the corresponding skill was considered as advanced.

Regulatory skills Regarding the regulatory strategies students reported they implemented, which were captured through the reflection prompts, they were analyzed qualitatively. Recall that to capture such strategies, we prompted students to reflect individually on their learning processes at the end of each project submission. The prompts aimed to identify what challenges students experienced in solving their projects, such as characterizations of self-monitoring, and the strategies they used to overcome such challenges, such as characterizations of cognition regulation, as shown in Table 6. Also, a team retrospective was written and submitted jointly by all team members along with each project submission. The team retrospective included prompts to characterize the challenges students experienced as a team and what strategies they implemented to improve the teamwork performance for their next project submission.

Although we primarily focused on self-monitoring and cognition regulation as two overarching inductive categories, our goal was to engage in deductive processes, so we describe in detail how these played out in the context of solving the scaffolded computational modeling and simulation projects.

These reflection prompts and team retrospectives were analyzed following a conventional content analysis approach, followed by a summative content analysis approach (Hsieh & Shannon, 2005). That is, during the conventional content analysis approach, we first generated codes derived directly from the text data, and then those codes were compared, contrasted, and logically organized, resulting in a set of categories with meaning (Vaismoradi et al., 2016). In this stage of the analysis, we identified exemplars of representative quotes of the categories (Braun & Clarke, 2006). Then, the categories were summatively counted and visualized to identify and report overall patterns (Hsieh & Shannon, 2005).

Researchers' positionalities

The positionality of the researchers is highly interdisciplinary, multicultural, and diverse. The research team consisted of (a) three females, two of them being faculty members and, another one being a graduate student, and (b) two males, both of them graduate students. The expertise of the team is a combination of engineering education research, computational science and engineering, and biomedical, mechanical, and computer engineering. The team also represents different academic positions, such as education researchers, engineering educators, faculty, and graduate and undergraduate students learning about research. Finally, our multicultural background also brings international perspectives from Africa, Asia, Latin America, Europe, and North America. The identities and roles, as well as the diversity in the team, have been reflected in the decisions implemented in the learning design and the research process.

Specifically, in the design process, our experiences as instructors or faculty and our own experiences as students in different countries informed the scaffolding design to be human-centered and culturally sensitive. In the research process, our expertise was critical, as the domain is highly complex and interdisciplinary. Thus, the analysis of the data required a high level of collaboration and learning among the team members. Finally, our multiple roles as faculty, students, and researchers allowed

Table 5 Scoring rubric for the projects for evidence of representational competence

Skill	Basic (1 point)	Proficient (3 points)	Advanced (5 points)
Computational model (code working properly, code organization)	Code often results in errors OR Code often produces extremely inaccurate results	Code is largely error-free, and results are relatively accurate AND Code is not organized with repeating sections of code	Code is largely error-free, and results are relatively accurate AND Code is highly organized and has a logical flow (sensible variable names, not a lot of repeating structures, etc.
Meaning-making of the code (in-code comments focused on the science vs. programming)	Code comments are largely missing from the project	Coded comments are present but extremely basic and largely focus on explaining WHAT the student did	Coded comments are detailed and largely focus on explaining WHY decisions were made
Graphical representations (proper graphical code outputs)	The graph code fails to make some graphs OR Graph code often creates highly inaccurate graphical results OR No graph code was provided, and a highly inaccurate graph outlook	Graph code makes all graphs but with some inaccurate headings and axis titles and insufficient axis plot intervals	Graph code makes all graphs with accurate headings, axis titles, and sufficient axis plot intervals
Explanations and arguments of output (proper claim with no evidence, proper claim with evidence)	Explanations are mostly very basic (one sentence, missing, argumentation, etc.) OR Some explanations are inaccurate or vague claims	Explanations are more in-depth but miss some argumentation for the student's response OR Some explanations are very basic, and some are very detailed	Explanations are very detailed and often give an argument for why the answer is given

Table 6 Categories for characterizing learning challenges and strategies to address them

Regulatory skills	Definitions
Self-monitoring	Instances involving students assessing their learning and learning processes in the form of experienced learning challenges individually and as a team
Cognition regulation	Instances where students mentioned applying critical thinking skills, such as searching and use of alternative solutions, and pulling together the acquired information from different sources

us to better protect students' privacy and confidentiality by establishing procedures where the course instructor would not be involved in the analysis but would inform the interpretation of the findings as a whole. The course instructor also brought in the practical perspective of what is feasible in the classroom and the learnings needed to improve the learning experience for the students in future implementations.

Our roles in the project were as follows: a senior discipline-based education researcher worked with the faculty member who was also the course instructor on revising the projects to incorporate the scaffolding and plan for the data collection. The initial discussions determined what was feasible in the context of a semester in terms of the number of projects, the amount of work required for each project, and additional work accommodating for data collection. Once the plan was formed, the senior researcher worked with two of the graduate students. One graduate student had just completed his dissertation investigating computational thinking in the context of modeling and simulation in engineering education, and the second graduate student, with a programming and data science background, had just started her doctoral program. The initial team worked together to revise the projects, ensuring they were technically and pedagogically sound. A third graduate student with a background in mechanical engineering and a deep understanding of thermodynamics joined the team once the data was collected. His disciplinary expertise and the expertise of the senior graduate student in biological engineering were essential to the data analysis procedures. The undergraduate student assisted with data processing, data analysis, and data visualization tasks.

Ethical and trustworthiness considerations

Before initiating the research activities, we obtained an Institutional Review Board approval from our university. The study was approved as exempted by the Institution's Review Board under protocol No. IRB-2021–1702, since it involved research conducted in the classroom that specifically involves normal educational practices that are not likely to adversely impact students' opportunity to learn or the assessment of educators. Specific ethical considerations consisted of ensuring students' privacy and confidentiality. For this, we conducted the data analysis once the semester was completed and the grades were submitted. Also, the data analysis was conducted by the discipline-based education researchers in the team, leaving out the course instructor at this stage.

Multiple strategies were implemented to ensure the study's trustworthiness. To ensure the study's credibility, we implemented triangulation processes through three rounds of data collection within the same semester, where multiple data sources and repeated measures helped us identify the observed patterns (Stahl & King, 2020). To promote the study's transferability (Gunawan, 2015), we provided thick descriptions of the projects, the deployment of the projects within the course, and samples of students' work from at least one project where observations and explanations from two teams were used to inform the process of creating and refining the scoring rubric. Finally, the reproducibility of the study was ensured by a consensus-making process between two researchers (Watts & Finkenstaedt-Quinn, 2021).

Two researchers followed a consensus-making process to approach the analysis of the project data regarding the identification of computational model-based learning and representational competence. For this, the first researcher, trained in qualitative analysis, generated the first version of the chunking (and coding) of the observations for computational model-based learning within a subset of the data consisting of ten students' submissions corresponding to Project 1. Based on this initial analysis, the second researcher generated an initial version of the rubric to score representational competence guided by samples of rubrics from previous research and iteratively refined it according to observations found in the actual data. When an initial rubric was developed, the first researcher performed an audit trial by testing the rubric on a subsample of the data the second researcher already scored. In this way, the two researchers engaged in an iterative process of scoring the data, improving the scoring rubric, and coming together to validate their observations. The second researcher then, after a waiting period of time, recoded the same data to ensure dependability. Once the second researcher was satisfied with the recoding process, he proceeded to score the rest of the entire dataset by himself, but regularly consulting with the first researcher when unsure of how to code specific observations.

Two other researchers followed a similar consensus-making approach to analyzing the reflection data that characterized the regulatory skills in terms of the

experienced challenges and the applied strategies for addressing the challenges. However, instead of chunking the data this time, the first researcher established initial codes with corresponding definitions. These codes were shared with the second researcher, who coded the same initial subset. The researchers then met to discuss initial codes and revised them based on discrepancies. The two researchers decided on the final application of the coding scheme for the entire dataset. Then, the two researchers divided the data into about equal parts and coded it independently. Once they coded the full dataset separately, the two researchers came back together once again and jointly defined the categories. Once the codes and categories were fully analyzed, tables were used to organize the data into teams in order to report final insights. The selection of the representative quotes was also done in collaboration between the two researchers to ensure the selected quotes truly characterized the definition of the category.

Results

The results are organized into three main subsections. The first sub-section provides in-depth descriptions of how computational model-based learning resulted from teams of students engaging in solving the scaffolded computational modeling and simulation projects. The second sub-section provides the results of students' levels of representational competence, and the third subsection delivers qualitative descriptions of the regulatory skills students reported that were applied to support their learning processes and team coordination processes.

Computational model-based learning

In the first stage of the analysis, our goal was to characterize how students working in teams evidenced aspects of computational model-based learning. Through our qualitative analysis, we identified two major groups. One group ($n=16$) consisted of Teams 2, 5, 8, 10, 11, 12, 13, 14, 16, 17, 19, 20, 21, 22, 23 and 24 whose solutions evidenced a meaningful engagement with computational model-based learning, and another group ($n=8$) consisted of Teams 1, 3, 4, 6, 7, 9, 15, and 18 whose solutions evidenced an engagement with computational model-based learning more superficially. To exemplify differences between the two groups of teams, we purposefully selected artifacts produced by two teams: Team 1 as an example of a team that engaged with computational model-based learning more superficially, and Team 2, as an example of a team that demonstrated a more meaningful engagement with computational model-based learning.

To demonstrate the range of possible outcomes generated by the two groups of teams, this sub-section is organized into four main stages describing how computational model-based learning resulted from students' engagement with the scaffolded computational modeling and simulation projects. For this, we focused on Project 2, which deals with students creating models to analyze a set of gene regulatory modules for the purpose of neutralizing extracellular toxin at varying production rates. Once the models were created, a second goal was for the students to implement the model in a computer-programmable environment to characterize and describe the biological interactions and system dynamics. To approach their solutions, students had to (a) determine a system of differential equations describing the dynamics of the biological system and build a corresponding model, (b) use the computational model to predict what the dynamics of receptor, toxin, and anti-toxin levels are over time, (b) improve their models by including one or two regulatory modules for activation of anti-toxin production, and (c) optimize the system by finding a combination of to three gene regulatory modules that can meet the design criteria.

Analyze the problem, plan, and formulate the model. For this portion of the project, students were guided through the initial stages of the modeling process to identify a preliminary procedure or plan. The guidance took the form of process management scaffolding that intended for students in a team to consider and recall the disciplinary knowledge they would need to approach the project solution and consider an initial approach for solving the problem. Figures 5 and 6 depict two samples of planning approaches provided by two different teams on Project 2.

While the planning approach for Team 1 focused on using a flowchart to describe the implementation of a solution model at the code level, Team 2 focused on formulating a stepwise procedure required to solve sets of coupled differential equations that represent a solution model plan. Team 1's flowchart illustrates the process workflow with a decision point that depends on the transient behavior of the represented biological phenomenon. In Fig. 6, Team 2 considered obtaining numerical solutions based on the model's initial conditions and included an expectation to generate new output plots from potential outcomes of planned optimization and sensitivity analysis.

Team 1's flowchart-based plan provides the students with a high-level conceptual view of the solution model, while Team 2's algorithm-based plan presents the precise mathematical framework that includes ordinary differential equations (ODEs) and execution steps for the modeling and simulation plan. A reflection from a student in Team 2 suggests some merit to the team's algorithm-based plan when asked about the project's most

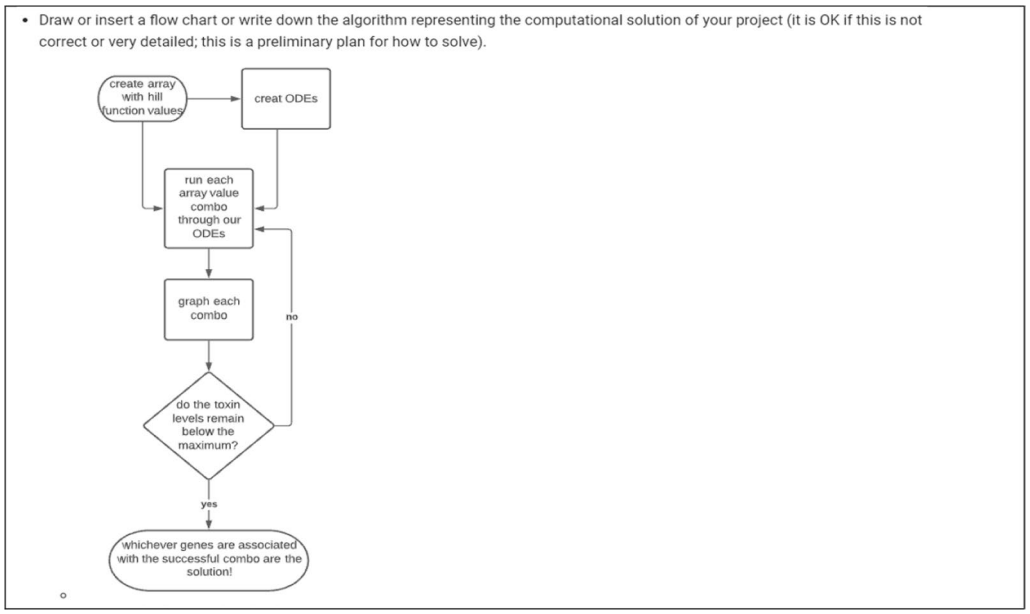


Fig. 5 Team 1's plan for approaching the problem in the form of a flowchart

- Draw or insert a flow chart or write down the algorithm representing the computational solution of your project (it is OK if this is not correct or very detailed; this is a preliminary plan for how to solve).

Create the differential equations using the parameters required to determine the rate for the toxin, receptor, anti-toxin, and receptor-toxin complex:

Toxin (T): $\frac{dT}{dt} = k_{toxProd} - k_{toxDecay}[T] - k_{neut}[A][T] - k_{on}[R][T] + k_{off}[RT] + k_E[RT]$

Receptor (R): $\frac{dR}{dt} = beta_2\left(\frac{[RT]^{n2}}{[RT]^{n2}+K_1^{n2}}\right) + beta_R - k_{on}[R][T] + k_{off}[RT] - k_E[R]$

Anti-toxin (A): $\frac{dA}{dt} = beta_1\left(\frac{[RT]^{n1}}{[RT]^{n1}+K_1^{n1}}\right) - k_{antiDecay}[A] + k_{neut}[A][T]$

Receptor-Toxin complex (RT): $\frac{dRT}{dt} = k_{on}[R][T] - k_{off}[RT] - k_E[RT]$

Using a code, numerically solve and plot the solutions with initial conditions.

Update the differential equation of the components of the system to incorporate the addition of gene regulatory modules with the new parameters of beta 1/3, beta 2, KI2, and n2 and updating the values of K1 and n1 from gene 1AT2 and 2AT1:

Toxin (T): $\frac{dT}{dt} = k_{toxProd} - k_{toxDecay}[T] - k_{neut}[A][T] - k_{on}[R][T] + k_{off}[RT] + k_E[RT]$

Receptor (R): $\frac{dR}{dt} = beta_R - k_{on}[R][T] + k_{off}[RT] - k_E[R]$

Anti-toxin (A): $\frac{dA}{dt} = beta_1\left(\frac{[RT]^{n1}}{[RT]^{n1}+K_1^{n1}}\right) - k_{antiDecay}[A] + k_{neut}[A][T]$

Receptor-Toxin complex (RT): $\frac{dRT}{dt} = k_{on}[R][T] - k_{off}[RT] - k_E[RT]$

Using a code, numerically solve and plot the solutions with initial conditions for a toxin production rate of 0.5 and 8 M/s.

Find a combination of up to 3 regulatory modulus that produce values that meet the criteria. Consisting of 1 or 2 modulus activation antitoxin production AND/OR 1 module activating receptor production. Possibly use 1AT1, 1AT2 and 2AT2 with their respective K and n values:

Toxin (T): $\frac{dT}{dt} = k_{toxProd} - k_{toxDecay}[T] - k_{neut}[A][T] - k_{on}[R][T] + k_{off}[RT] + k_E[RT]$

Receptor (R): $\frac{dR}{dt} = beta_R - k_{on}[R][T] + k_{off}[RT] - k_E[R]$

Anti-toxin (A): $\frac{dA}{dt} = beta_{1a}\left(\frac{[RT]^{n1a}}{[RT]^{n1a}+K_{1a}^{n1a}}\right) + beta_{1b}\left(\frac{[RT]^{nb}}{[RT]^{n1b}+K_{1b}^{n1b}}\right) - k_{antiDecay}[A] + k_{neut}[A][T]$

Receptor-Toxin complex (RT): $\frac{dRT}{dt} = k_{on}[R][T] - k_{off}[RT] - k_E[RT]$

Plot the toxin levels as well as incorporate print statements to show that the system meets the criteria of a max toxin production of below 0.35 for more that 90% of the time points for a range of production rate between 0.5 and 8 M/s

Fig. 6 Team 2's plan for approaching the problem in the form of differential equations and explanations

beneficial aspects. The student noted that *"Being able to see how each of the different steps of the project affected the outcome made it easier to understand how the different parts of the system affected each other over time."*

While a detailed flowchart was not an explicit requirement for this project, Team 1's flowchart lacked important workflow components like input model parameterization, initial conditions, result outputs, decision points for model testing, and validations (sensitivity analysis). These omissions may potentially limit the flowchart's ability to effectively convey the project's methodology sequence to students. Team 2's algorithm-based plan may also have posed a daunting challenge to students who might have been averse to outlines of mathematical representations and who did not have any form of visual or scaffolded outlook on the solution procedures. Overall, formulating a modeling plan of action by embedding the mathematical equations into the flowchart may better support students in building clear and concise visual models of the problem and solution workflow.

Implement and evaluate the model to solve the problem. Comparing Teams 1 and 2 model implementation approaches along with the respective impacts on solution models for one of the project tasks is observed in Figs. 7

```
#create a list of all the initial conditions
y0  = [Toxin0, ReceptorToxin0, Receptor0, Antitoxin0]

z   = 100                      #number of time steps
t   = np.linspace(0, 100.0, z)   #array for time step values from 0 to 100 seconds

##create function to solve a set of ODEs
def model(y, t):
  #labeling where we will track each species in the model output
  T       = y[0]
  RT      = y[1]
  R       = y[2]
  A      = y[3]

  #Mass Balance Equations

  #toxin
  dTdt  = k_toxProd + k_E * RT + k_off * RT - k_ToxDecay * T - k_on * T * R - k_neut * T * A
  #receptor-toxin complex
  dRTdt = k_on * R * T -  k_off * RT - k_E * RT
  #receptor
  dRdt  = (beta_2 *((RT ** n2)/((RT ** n2) + (K2 ** n2)))) + k_off * RT  + beta_R - k_E * R  - k_on * R * T
  #anti-toxin
  dAdt = (beta_1 *((RT ** n1)/((RT ** n1) + (K1 ** n1)))) -  k_antiDecay * A - k_neut * A * T

  #return a list that includes the increment values of the differential equations
  return [dTdt, dRTdt, dRdt, dAdt]

#using Euler's Method integration to output a solution
soln  = odeint(model, y0, t)

print(soln)

#create solution matrix with assigned variable names
Toxin            = soln[:, 0]
ReceptorToxin    = soln[:, 1]
Receptor         = soln[:, 2]
Antitoxin        = soln[:, 3]

#formulate an array for the toxin threshold
MaxAllowableToxin = [.35]*z

#plot all timesteps for species concentration
plt.figure(num=2)
plt.plot(t, Toxin,           label='Toxin')
plt.plot(t, ReceptorToxin,   label='RT')
plt.plot(t, Receptor,        label='Receptor')
plt.plot(t, Antitoxin,       label='Antitoxin')
plt.plot(t, MaxAllowableToxin, label='MaxToxin')

#plt.ylim([0,20])

#formatting and defining attributes of graph
plt.xlabel('Time (s)')
plt.ylabel('Species Concentration (M)')
plt.title('System Timeseries')
plt.legend(loc=1, bbox_to_anchor=(1.2, 1))
```

Fig. 7 Team 1's model implementation and corresponding code comments explaining the model

```
y0  = [Toxin0, ReceptorToxin0, Receptor0, Antitoxin0] #Initial concentration

z   = 1000 # Number of time steps
t   = np.linspace(0, 100.0, z)   # Time vector in seconds

# Defines a function that returns the differential equations for each of the components in the system
def model(y, t):
  # This code section assigns the row of the array to the corresponding component variable
  T       = y[0]
  RT      = y[1]
  R       = y[2]
  A      = y[3]

# This section of code defined the differential equations for the change in concentration of each component in the system
  dTdt  = k_toxProd[i] - k_ToxDecay * T - k_neut * T * A - k_on * T * R + k_off * RT + k_E * RT

  dRTdt = k_on * T * R - k_off * RT - k_E * RT

  dRdt  = beta_R + ((beta_2 * RT**n2) / (RT**n2 + K2**n2)) - k_E * R - k_on * R * T + k_off * RT

  dAdt = ((beta_1 * RT**n1) / (RT**n1 + K1**n1)) - k_antiDecay * A - k_neut * A * T

  # Returns the concentration of each component at each time step in an array
  return [dTdt, dRTdt, dRdt, dAdt]

# This is a loop that calculates the concentrations for each component of the system at time t for the toxin production rates of 0.5 and 8 M/s and plots them
for i in range(len(k_toxProd)):

  # Calls the defined funtion and solves the differential equation
  soln  = odeint(model, y0, t)

  # This code section assigns the row of the array to the corresponding component variable
  Toxin            = soln[:, 0]
  ReceptorToxin    = soln[:, 1]
  Receptor         = soln[:, 2]
  Antitoxin        = soln[:, 3]

  # This code section calculates the maximum allowable toxin concentration at time t
  MaxAllowableToxin = np.linspace(0.35, 0.35, z)

  # This code section plots the concentration of each component at each time ste
  plt.figure(num=i)
  plt.plot(t, Toxin,           label='Toxin')
  plt.plot(t, ReceptorToxin,   label='RT')
  plt.plot(t, Receptor,        label='Receptor')
  plt.plot(t, Antitoxin,       label='Antitoxin')
  plt.plot(t, MaxAllowableToxin, label='MaxToxin')

  # This code section applies Applies graph labels, titles, and legends
  plt.xlabel('Time (s)')
  plt.ylabel('Species Concentration (M)')
  plt.title(f'System Timeseries Rate with {k_toxProd[i]} (M/s)')
  plt.legend(loc=1, bbox_to_anchor=(1.2, 1))
```

Fig. 8 Team 2's model implementation and corresponding code comments explaining the model

and 8. The figures show how the teams began the model solution implementation in a Python code environment where initial values and input parameterization were set, along with an array of evenly spaced time vectors and a number of time steps used for the simulation. The teams followed the next steps by defining a function that takes in two arguments representing concentration values of selected biochemical components used to solve the sets of coupled mass balance ODEs. The teams went further to introduce the "odeint" numerical technique in Python's SciPy library that allows for a stepwise approximation of unknown functions to compute and store the first incremental solutions to the ODEs.

The initial solution sequence used by both teams was overall similar because of the scaffolded code templates provided for the project. However, Team 2 proceeded to use a 'for loop' to iterate through each biochemical component over time to calculate varying toxin production rates and, calculate allowable thresholds, and generate results in the form of graphical representations. Team 1's implementation appears to lack the 'for loop' essential for iterating through the biochemical components, hence limiting the model's flexibility and analytical capabilities. In contrast, the presence of the "for loop" in Team 2's code enabled students to systematically study the various gene modules while considering the impact of varying toxin production rates. That is, Team 2's model promoted a more efficient and organized approach to parameter exploration, which was suitable for robust sensitivity analyses and optimization studies compared to Team 1's model, which required duplicating functions to handle computational experiments.

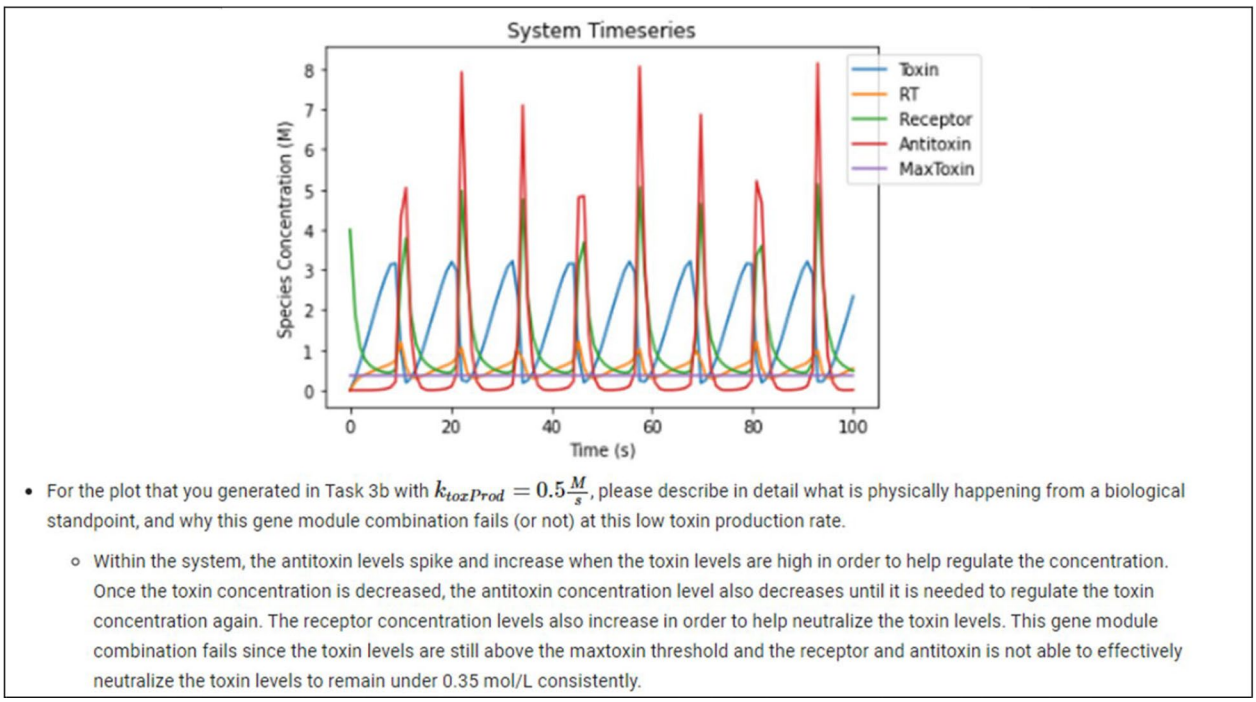
In Fig. 7, it can be observed that Team 1 employed an inaccurate step value of 100 over a 100-s time vector

range, resulting in a relatively large simulation time step value of 1.01 s. In contrast, Team 2 used 1000 steps over the same 100 s range leading to a finer simulation time step of 0.1 s. This discrepancy explains the errors in Team 1's model output results, specifically with the anti-toxin profile levels. Team 1's choice of a larger time step (1.01 s) resulted in some loss of important details when capturing rapid transient changes in the system, as seen in Fig. 9. The instructors for the project recommended a specific time step size value of 1001 and emphasized that it should not be less than 1000 steps. Regrettably, Team 1 did not adhere to this recommended step size in their implementation.

In the final stages of implementing the model using code, both teams used the provided code templates for extracting values from the solution matrix and creating the necessary plots. However, each team made some slight adjustments to the graph generation code. While these modifications did not impact the computational results produced by the implemented model, they did have a noticeable effect on the appearance and features of the generated graph plots. These adjustments primarily influenced aspects, such as the axis scale size, plot line colors, titles, axis labels, and plot legends. Team 1's modifications to the graph code resulted in plots that lacked proper title text and axis labels, which is an essential component for understanding the content and context of the graphs. This omission could have potentially made it more challenging for viewers to interpret the visualized data accurately. In contrast, Team 2's adjustments to the graph code were more successful in producing graph plots with accurate and expected features like titles, labels, and legends, enhancing the clarity and interpretability of the graphical representations of the model's results.

Model use and interpretation of the model outputs The explanations offered by Team 1 and Team 2 in Figs. 9 and 10, respectively, share similarities in their attempts to shed light on the behavior of gene module combinations under low toxin production rates. Both explanations agree on the system's oscillatory nature and its inability to stabilize at low toxin levels. Furthermore, they provide biologically grounded interpretations, emphasizing how concentrations of toxin, anti-toxin, and receptors interact to influence system dynamics. Nonetheless, as mentioned earlier, the discrepancy in the computational time step of Team 1's model is a crucial factor contributing to the visible mismatch in spike levels observed in the anti-toxin profile. This mismatch becomes evident when comparing Team 1's model to Team 2's model and the actual solutions of the system model.

It appears that the students in Team 1 did not fully grasp the significance of the time step error as a potential contributor to the model's output discrepancies, as this factor is not explicitly explained by the students in Fig. 9. This lack of comprehension can be attributed to two key



- For the plot that you generated in Task 3b with $k_{toxProd} = 0.5\frac{M}{s}$, please describe in detail what is physically happening from a biological standpoint, and why this gene module combination fails (or not) at this low toxin production rate.
  - Within the system, the antitoxin levels spike and increase when the toxin levels are high in order to help regulate the concentration. Once the toxin concentration is decreased, the antitoxin concentration level also decreases until it is needed to regulate the toxin concentration again. The receptor concentration levels also increase in order to help neutralize the toxin levels. This gene module combination fails since the toxin levels are still above the maxtoxin threshold and the receptor and antitoxin is not able to effectively neutralize the toxin levels to remain under 0.35 mol/L consistently.

Fig. 9 Team 1's model result output and interpretation

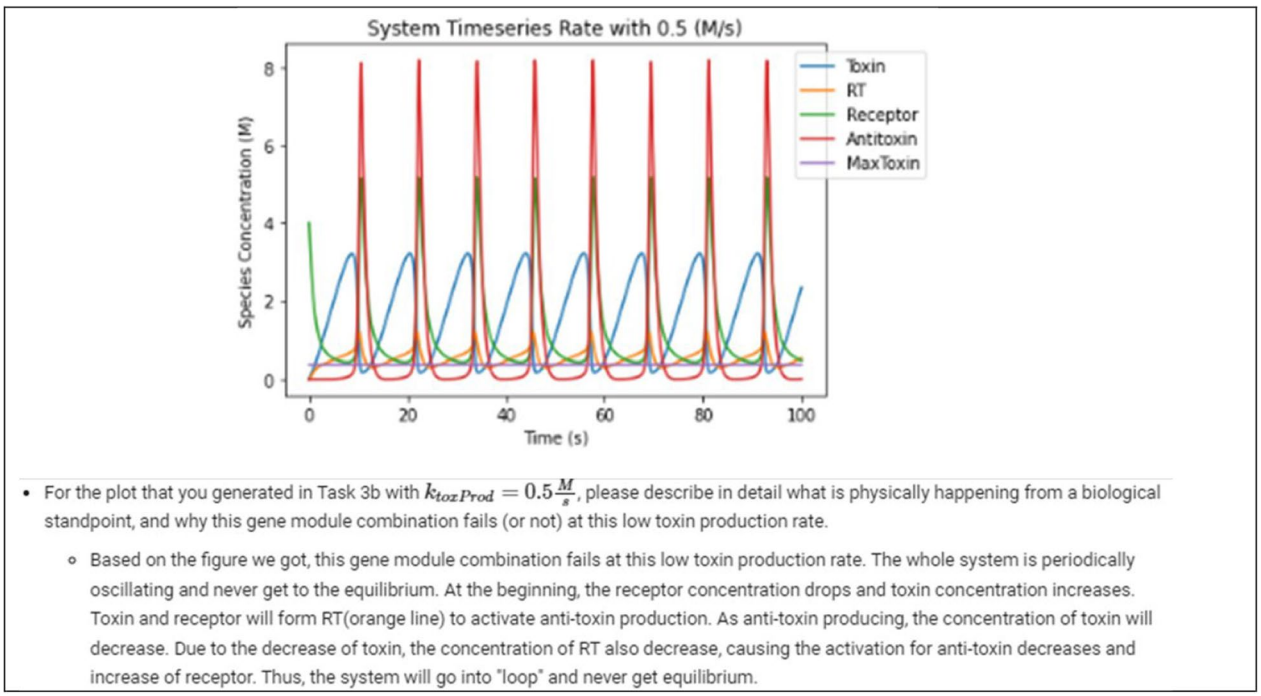


Fig. 10 Team 2's model result output and interpretation

reasons. First, the students may not have an in-depth understanding of the fundamental biomechanics underlying the system's dynamics. To know what's inaccurate, one needs to know what's accurate by having a solid grasp of the underlying principles. To underscore the importance of the students' background knowledge, it's notable that both teams in their result discussions omitted to cite the influence of a significant underlying variable, known as the hill function term. In the students' reflections below, some students have raised these concerns:

> *Student A: "My biggest struggling was understanding where hill functions fit within the differential equations and how to correctly incorporate them"*
> *Student B: "I think the most challenging thing I met during the project is to reflect "whether our model" is successful or not. To be more specific, after making sure that our code is correct, the results of our models are not that "ideal". Thus, this took a lot of time to think about it."*

Second, the students may not have acquired sufficient expertise in computational modeling and simulation to monitor the numerical error margin during the convergence process and to recognize the trade-offs between model accuracy and computational costs (memory consumption and computational speed) during the simulation time. In this case, the student's computational simulation experience holds paramount importance because, in situations where students may identify the plot errors in Fig. 10, they might still struggle to connect these errors to the time step parameter in the code, especially when the code execution appears to be successful. The ability to bridge the gap between the observed discrepancies and the underlying computational parameters is a crucial skill in ensuring the accuracy and reliability of computational models.

Report on the solution of the problem and reflect on the process While there was not a formal report for the project solution, as the computational notebook that embedded the indications and guidance was the actual final deliverable, students had an opportunity to reflect on the process. Students reflected individually on their learning and their team processes collectively. The samples selected for reflecting on the process for Project 2 focused on students' individual reflections on their learning. Figures 11 and 12 present the individual reflections for Team 1 and Team 2, respectively, regarding their learnings in Project 2.

In terms of skills acquired by members of Team 1 and Team 2, both groups reported that they had to develop a range of skills to accomplish the project. While there are some similarities in the skills mentioned, such as improving python coding skills and working with equations through building differential equations, mass balance equations, or translating biological systems into mathematical models, there are also differences in the specific

(2 points) Please answer the following questions *individually*. Mark your contribution with your name next to it.

- What skills did you have to develop to accomplish this project?
  - J: During this project I learned how to create my own differential equations from varying gene systems as well as learned how to implement a hill function into python coding.
  - F: This project furthered developed my Python coding and taught me how to code equations. I also learned more about mass balances for transcription networks.
  - K: In order to accomplish this project, I had to improve my skills in code debugging. I learned how to step through the different sections of code in order to find what is causing the mistake in our output.
  - A: During this project I learned how to further calculate differential equations and mass balance equations. I also learned how to plot timesteps within Matlab using Euler's Method. I had to refine my knowledge in hill functions and differential equations in order to complete this project.
  - E: I had to develop skills in working with Euler's method in the context of ordinary differential equations. Also, I had to work to understand how to translate the biological system into math modeling.
- What aspects of this project were the most beneficial for your learning?
  - J: The furthering of my python skills including: integration, hill functions and more complicated plots was the most beneficial to my learning.
  - F: Task one was the most beneficial for me since it taught me a lot about the mass balance equations. I quickly learned that without these we could not complete the project.
  - K: This project was very detailed, and that benefitted my learning because it taught me to be able to step back and view the entire problem while still understanding individual pieces.
  - A: The step-by-step process of this project was beneficial for me since it helped my comprehension of the project as a whole and how engineers go about solving rela world problems in a project based setting.
  - E: I think taking the problem all the way from biological concepts to math modeling to code was the most beneficial part of the process for me.

Fig. 11 Team 1's individual reflections on team members' learnings

skills emphasized by each team. Team 1 focused mainly on technical and mathematical skills. They mentioned developing proficiency in creating differential equations, implementing functions in Python, debugging code, and using mathematical methods like Euler's Method. Team 2, on the other hand, highlighted a broader set of skills. They mentioned communication skills, patience, and the ability to understand and translate code from pseudocode into Python. Therefore, while Team 1 prioritized the technical aspects of the project, such as mathematical modeling and coding proficiency, Team 2 recognized not just the technical aspects but also the interpersonal and problem-solving skills. This further highlights the complex nature of these computational modeling projects for students working in teams.

Regarding the aspects of this project that were most beneficial for students' learning, both teams reported gaining valuable insights from their experiences in mostly similar and some different ways. Both groups emphasized the significance of practical, hands-on learning. They mentioned the value of seeing how changes in code and parameters affected the project outputs. They both appreciated the visual learning aspect, with students noting the importance of graphs, plots, and visualizations for their understanding. They also both expressed an appreciation for the project setup structure in the computational notebooks that guided them through the progression of the project from simpler to more complex aspects, which enhanced the building of foundational knowledge and end-to-end holistic project understanding. With no stack difference between the two teams, some of Team 1's responses focused on the technical and coding aspects of the project, like the development of Python skills, integration, hill functions, and code execution, while Team 2's responses also highlighted the importance of visualization and conceptual understanding of the biological model beyond the coding aspect.

Representational competence

Representational competence was operationalized as students' ability to connect the computational representations of their models, either in the form of code or plots, to biological phenomena. For this, the rubric presented in Table 4 was used to score the accuracy of the models the students generated as the solutions for each of the three projects implemented throughout the semester and students' explanations of such models and their outputs. The first two criteria were aligned with the computational learning objectives of the course, which focused on creating computational models and students' ability to recognize the behavior of such models by explaining their functionality via in-code commenting. The second two criteria were aligned with the disciplinary learning objectives of the course, focused on generating the

(2 points) Please answer the following questions *individually*. Mark your contribution with your name next to it.

- What skills did you have to develop to accomplish this project?
  - E: I had to review python skills for using f strings in print statements. I also had to develop better communication skills in order to express my thoughts clearly so that everyone can undertand what I am thinking.
  - P: I had to recall how to build equations from reactions given reaction rates. I also had to be more patient and allow others to contribute so we all learn from the project.
  - N: Throughout this project, I had to develop my understanding and translation of code from pseudocode into Python. I have the least experience with Python out of the recent coding languages used, so I needed to review my skills to adequately contribute to my group members work.
  - G: Python and coding are not one of my strengths and so in order to help determine the new method of how to find the best combination, I had to learn and understand from my peers code how the langauge was transcribed in order to come up with solutions that could be applied and utilized.
  - X: In this project, I am more familiar with the Python selection and loop structure. Have a better understanding on (1)how gene regulation happens (2)how the parameters in Hill function influence gene regulation.
- What aspects of this project were the most beneficial for your learning?
  - E: The aspects of this project was that I was able to try different things in the code to see if what I thought would happen with changing different rates was true. Being able to see how each of the different steps of the project affected the outcome made it easier to understand how the different parts of the system affected each other over time. I am a visual learner and have to see what happens and be able to write it out to learn it. I was able to do that for this project.
  - P: Graphing and printing results is very helpful for me to trace what exactly is happening, and whether the model output makes sense. By plotting at different concentrations, I was able to troubleshoot problems that were not readily obvious at just one concentration, and thus build a more robust model.
  - N: Being able to simulate throughout different parts of the code the changes that then we would apply in the last part to choose the gene was very helpful and beneficial in my understanding of this project. This helped to establish a baseline that would be used to synthesize in the next steps.
  - G: The progression of the parts in the project that started with a simpler execution and building into more complex. This for me was beneficial because I could see the skeleton of the basic code that applied the equations and see how the code built up into more complex in order to compelete more tasks and compare more data.
  - X: The most helpful thing I learned from this project is having a better understanding of the gene regulation model. From what I have learned in class, the change of concentration of different components in a gene regulation model is quite abstract. After finishing this project, I have a more specific and clear idea on how things happen.

Fig. 12 Team 2's individual reflections on team members' learnings

proper solution to the biological problem in the form of plots and interpreting those accurately via explanations. Specifically, we identified and measured that the computational model executed and worked properly, the graphical representations or output were accurate, and that their explanations of the behavior of the system in the form of in-code commenting were accurate, as well as the explanations of the output of the computational model regarding students' interpretations.

Table 7 presents the descriptive statistics for each of the three projects organized by the four constructs of a computational model, graphical representations, meaning-making of the code, and explanations of output. As can be observed, overall, students performed at the advanced level in terms of the ability to create a computational model representing a biological system. However, the ability to properly create graphical representations is overall more consistent with a proficient level of achievement. Regarding the understanding of the code in terms of in-code comments and the explanations of the interpretations of the output of the computational models, the student's achievement was at an

Table 7 Projects' performance scores for evaluating representational competence

	Project 1		Project 2		Project 3	
Construct	**Mean**	**Std. Dev**	**Mean**	**Std. Dev**	**Mean**	**Std. Dev**
Computational model	4.66	0.74	4.16	1.14	4.25	0.96
Meaning-making of the code	4.41	1.07	4.0	1.0	3.91	1.41
Graphical representations	4.16	1.14	3.16	0.98	3.75	1.80
Explanations of output	4.41	0.90	4.0	1.0	3.83	0.98

advanced level for Project 1 but at a proficient level for Projects 2 and 3.

We performed a second analysis at the team level to explore if the teams had similar patterns in their performance regarding (a) the implementation of the model and the understanding of the inner workings of the model via in-code comments and (b) the use of the model to solve the biological problem and the explanations of the outputs of the model. Table 8 presents the average scores for each of the criteria across all the projects and the overall team performance throughout the semester.

We identified performance patterns where Teams 17 and 21 performed relatively higher across all projects and all criteria, while Teams 6 and 4 performed relatively lower across all projects and all criteria. We also identified that Teams 5, 8, 12, 13, and 14 performed relatively higher in the criteria focused on creating computational models and commenting the code properly but somewhat lower in criteria focused on using the models and interpreting the outputs of the model to solve the biological problem. In contrast, Teams 1, 15, and 19 performed relatively lower in the criteria focused on evaluating the computational models and explaining those via in-code commenting but relatively higher in criteria focused on using models and interpreting the outputs of those models to solve the biological problem. The rest of the teams performed on average across all criteria.

Regulatory skills

Regulatory skills were operationalized as self-monitoring, describing instances involving students assessing their learning and learning processes in the form of experienced learning challenges individually and as a team, and cognition regulation in the form of evidence of students' application of strategies aimed at overcoming learning challenges and teamwork strategies and skills for working with others. To characterize students' processes, we performed a qualitative analysis focusing on students' individual and team-based reflections submitted along with each project submission. Individual team members reported learning challenges, while collaboration challenges were reported as a team. Regarding self-monitoring regulation, we identified three main categories: (a) challenges and strategies related to programming skills, (b) challenges and strategies related to meaning-making skills for understanding and connecting the science to the code and the results, and (c) challenges and strategies related to process management mainly focused on

Table 8 Projects' performance scores for evaluating representational competence by team

Team	Computational model	In-code comments	Graphical representation	Explanation of output	*M*	SD
1	3.67	5.00	4.33	5.00	4.50	0.90
2	4.33	3.67	5.00	4.33	4.33	1.30
3	4.33	4.33	4.33	3.67	4.17	1.03
4	3.67	5.00	2.33	3.67	3.67	1.56
5	5.00	3.67	2.33	3.00	3.50	1.51
6	4.33	3.67	2.33	3.67	3.50	1.24
7	4.33	3.67	4.33	3.67	4.00	1.04
8	5.00	4.33	3.00	5.00	4.33	1.30
9	4.33	5.00	3.67	5.00	4.50	0.90
10	4.33	3.67	3.67	3.67	3.83	1.03
11	3.67	3.67	4.33	3.67	3.83	1.03
12	5.00	3.67	2.33	3.67	3.67	1.56
13	5.00	3.67	3.00	4.33	4.00	1.35
14	5.00	5.00	3.00	3.67	4.17	1.34
15	3.67	5.00	5.00	5.00	4.67	0.78
16	4.33	3.67	4.33	3.67	4.00	1.04
17	4.33	5.00	4.33	4.33	4.50	0.90
18	3.67	3.67	4.33	3.67	3.83	1.03
19	4.33	3.00	4.33	4.33	4.00	1.35
20	4.33	5.00	3.67	3.67	4.17	1.03
21	5.00	3.67	4.33	3.67	4.17	1.03
22	4.33	4.33	3.00	4.33	4.00	1.35
23	4.33	3.00	3.67	5.00	4.00	1.60
24	4.33	4.33	3.67	4.33	4.17	1.03

project management skills. Each of these challenges is described below, along with the cognition regulation strategies students implemented in order to overcome such challenges.

Programming challenges The first theme addresses the team's challenges regarding the programming requirements of the projects. Specifically, teams struggled with transforming the mathematical equations into computational representations, generating the required plots, learning the programming language's syntax, and troubleshooting and debugging the code. Some team members also experienced difficulties reading someone else's codes, especially when not commented. One student expressed their challenges with Python syntax: "One challenge for me was figuring out the necessary syntax to code things correctly in Python." Another one indicated the complexity of learning concepts simultaneously: "My biggest challenge was working through both the code and the overall thermodynamics because as the system got more complex, the equations became harder to solve and understand." Another student highlighted the challenges related to visualizing and plotting "I found the plotting of the functions to be the most challenging because the functions can be hard to visualize." The first four columns in Table 7 summarize the experienced programming challenges reported by team members. As can be observed in the patterns presented in Table 7, almost every team had difficulties deriving the equations and dealing with aspects of programming syntax.

To overcome the challenges, students implemented various cognition regulation strategies like seeking help from others through consultations with teammates, teaching assistants, or instructors or engaging in independent study by utilizing online resources, textbooks, and class materials. A student is quoted as seeking help from various resources: "I overcame the challenges I faced by asking my group members for help when I didn't understand and reading through the notes from lecture when there was confusion about how to begin finding a mathematical solution." Another student utilized the provided class resources, "The professor provided lots of resources that I could use to brush up on my Python skills and reference throughout the project," while other students consulted each other like this one "I overcame the problem by brainstorming with other students from my group and using a lot of research power on the internet to gain a deeper understanding of the concepts at work in this problem." Table 8 summarizes the strategies teams implemented to overcome their programming challenges. As can be observed in the patterns presented in Table 8, peer learning and feedback, attending office hours, and debugging code were the strategies implemented by almost every team.

Meaning-making skills The second theme addresses the team's difficulties in interpreting and reflecting on how the computational models work and what the model results mean. The theme is derived from some teams reporting challenges with understanding the engineering tasks asked and understanding how the overall model system works. This is seen in students' difficulty with interpreting the provided code blocks and writing in-code comments that connect engineering with programming. One student said, "The most challenges I met in the project is to understand what is going on in the system to explain the result from my code." Another explained the technical difficulties as follows: "I initially had some trouble finding how all of the components of the system came together into the mass balance equation. The biggest challenge for my task personally was understanding how the ODEs needed to be set up in order to accurately model the function of the enzymes." Another wondered about the relevance of the inputs to the problem task: "I struggled with knowing why different values input in Part 6 were significant in the context of the problem." Finally, this student explained the challenges of understanding the graphical outputs:

> *"The main challenges encountered in completing this project were related to taking a step back after our graphical displays were working properly to ask if they make sense. I needed to discuss with my teammates to understand why the shape changes when we leave a steady state assumption."*

The last three columns in Table 9 summarize the experienced meaning-making challenges reported by team members. As can be observed in the patterns presented in Table 9, the most common challenge experienced by team members involved generating the plots and understanding the underlying science needed to understand the system as a whole.

The cognition regulation strategies used to overcome the challenges detailed in the theme were similar to those used in the first theme about programming. For example, consultations with teammates, teaching assistants, or instructors or engaging in an independent study by utilizing online resources, textbooks, and class materials. Other students expressed that examining the code and analyzing how it works was beneficial, especially with this theme. One student reported, "The questions made me examine the code and dig into the WHY, which was beneficial to my learning. I went through each section and tried to understand the fundamentals." Another student said, "We had to find outside information as well as seek additional help from office hours as well as TAs. We required finding more insight into the specific parameters in order to understand how to analyze the graphs

Table 9 Self-monitoring regulation regarding programming and meaning-making challenges

Team	Programming challenges				Meaning-making challenges		
	Deriving equations	Understanding Python syntax	Understanding others' code	Troubleshooting or debugging	Generating the plots	Interpreting the outputs	Understanding the science
1		X					X
2	X		X		X	X	X
3	X	X	X		X	X	X
4	X	X		X	X		X
5		X	X		X		X
6	X	X	X				
7			X	X	X		X
8	X	X		X		X	X
9	X	X					X
10	X	X			X	X	X
11	X	X		X	X		
12	X	X		X	X	X	X
13	X	X	X	X		X	X
14	X	X		X	X	X	X
15	X	X		X	X		X
16	X	X			X	X	
17	X	X					
18		X	X	X	X		X
19	X	X		X	X		X
20	X	X		X	X		
21	X	X				X	X
22	X	X		X	X		X
23	X	X				X	X
24	X	X					X

and plots," and another reported that "By brainstorming with classmates and reading up on the topic myself for a better understanding." Table 10 summarizes the strategies teams implemented to overcome their meaning-making challenges. We noticed that the strategies teams implemented to overcome their programming challenges were similar or the same as those students implemented to overcome their meaning-making challenges, where peer learning and feedback and attending office hours were the most frequent strategies implemented by almost every team. This pattern is understandable as the experienced programming and meaning-making challenges were directly related to model-based reasoning processes where students had to create, use, evaluate, or interpret models, which are highly interconnected processes.

Process management skills The third theme addresses the students' difficulties in the process management of the project in terms of planning, communication, division of labor, time management, teamwork coordination, and using the Google Colab technology. Regarding coordination and time management, a student reported, "Not only was the coding itself difficult, but it was hard to work through our varying schedules and find the time for all of us to meet and complete the project." Concerning the division of labor, a student reported:

> *"I thought the most challenging part of this project was properly coordinating how the work could be distributed among five people with very different schedules. We struggled to find an appropriate meeting time, and I believe this may have reduced teamwork."*

Table 11 summarizes the challenges the teams experienced regarding their process management. It appears the most common issues reported by most teams were poor project management, clashing schedules to meet and engage in collaborative or individual work, and technical issues experienced with Google Colab.

The teams handled this challenge using various cognition regulation strategies like asynchronous coding in Google Colab at different times, scheduling regular asynchronous and synchronous meetings, assigning internal deadlines to tasks for each team member, discussing

Table 10 Cognition regulation strategies for overcoming programming or meaning-making challenges

Team	Utilized online resources	Attended office hours	Peer-teaching or team collaboration	Utilized the textbook or in-class materials	Performed independent study	Broke down tasks into chunks	Commented code, print statements, and trial and error
1	X	X	X		X		X
2		X	X				
3	X	X	X		X		
4	X	X	X	X			X
5	X		X	X	X		
6	X	X	X		X		
7			X				X
8	X	X	X	X			X
9	X		X				
10		X	X	X	X	X	
11		X	X			X	X
12	X	X	X				
13	X	X	X				X
14	X	X	X	X	X		X
15		X	X	X	X		X
16		X	X	X			X
17				X			X
18		X	X		X		X
19	X		X			X	
20			X	X	X		X
21		X	X				
22		X	X	X	X		
23			X	X	X		
24		X	X				

the division of labor, and increasing team collaboration. Regarding team coordination, a student reported:

> *"As the team coordinator for this project, I created a secondary document to track the progress of each section and suggested that the teamwork in order. Team members worked expeditiously to work through every part of the code so that the next person could start working on theirs."*

Regarding the division of labor, a student reported, "teamwork strategies that worked included separating different tasks for the people who were available." Table 12 depicts the strategies the teams reported implemented for overcoming process management challenges. Planning for the division of labor and improving communication processes were implemented by about half of the teams. Also, to overcome the technical challenges with Google Colab, students created schedules and rotations to work on the computational notebook asynchronously.

We further explored the data to identify possible patterns between teams' computational model-based learning, representational competence, and regulatory skills. No significant patterns were identified regarding commonly experienced challenges or typically implemented strategies related to performance between the meaningful or superficial forms of engagement with computational model-based learning. Overall, teams reported about seven experienced challenges and six strategies implemented to overcome challenges across programming, meaning making, and process management skills. We also explored if a relationship existed between the number of reported challenges and strategies and overall performance, and no significant patterns were identified. However, we offer some qualitative observations here. It appears that Teams 17 and 21, who overall performed at the advanced level across all three projects and were part of the meaningful engagement with the computational model-based learning group, reported experiencing an average number of challenges and fewer than the average strategies for overcoming the challenges they experienced. We also identified that teams that performed at the advanced level regarding the implementation of the model and the understanding of the inner workings of the model via in-code comments, but at the basic level

Table 11 Self-monitoring regulation regarding process management challenges

Team	Poor team communication	Poor project management	Clashing work schedules	Poor team coordination	No good division of labor	Technical issues with Colab
1			X			
2	X		X			
3						
4			X	X	X	X
5						X
6						X
7	X	X	X		X	X
8	X	X	X	X	X	X
9			X			X
10				X		X
11		X				X
12	X	X			X	X
13		X				X
14		X	X			X
15		X				X
16		X	X			X
17		X	X	X	X	X
18		X				X
19		X	X	X	X	
20		X	X			
21						X
22						X
23		X	X			X
24		X	X			X

regarding the use of the model to solve the biological problem and the explanations of the outputs of the model (i.e., Teams 5, 8, 12, 13, and 14), on average experienced more challenges, about nine, and reported implementing about eight strategies to overcome those challenges. These five teams were also in the group with teams that engaged meaningfully in computational model-based learning. The teams that were, on average, proficient across the board reported experiencing an average number of challenges, about six or seven, but mentioned they implemented about five strategies to overcome those challenges, which was lower than the average.

Discussion and implications

Our overall findings revealed that our learning design consisting of computational modeling and simulation collaborative projects scaffolded through d processes and delivered via computational notebooks resulted in enactments of computational model-based learning. Specifically, sense-making and process management scaffolding guided students through enactments of (a) analyzing the problem and planning models to approach the solution of the problem, (b) implementing and evaluating the models to solve the problem, (c) using the model and interpreting the outputs of the model, and (d) reporting on the solution of the problem and reflecting on the process. Such enactments are critical for students to promote model-based learning (Develaki, 2017; Lehrer & Schauble, 2006). However, based on qualitative observations from our characterizations of computational model-based learning, students experienced these processes differently, even when the scores of their representational competence were comparable. Specifically, while one group of 16 teams, such as Team 2, engaged more meaningfully in the computational model-based learning process, another group of 8 teams, such as Team 1, appeared to be less engaged in in-depth model-based reasoning. Thus, this finding suggests that some teams may need more scaffolding or feedback through the modeling and simulation cycle.

A second outcome of the study focuses on ways to support students' development of representational competence. Specifically, articulation scaffolding supported students in their computational model-based

Table 12 Cognition regulation strategies for overcoming process management challenges

Team	Regular asynchronous time	Assigned internal deadlines	Planned division of labor	Improved communication processes	Improved team cohesion	Tracked peers' progress	Setting schedules for Colab edits
1	X		X	X			
2				X			
3							
4		X	X	X	X	X	X
5			X				X
6							
7	X	X	X				X
8		X	X	X	X	X	X
9			X				X
10	X			X			
11			X	X	X	X	
12	X	X	X	X			
13			X	X			X
14	X						X
15	X	X	X	X			X
16							X
17	X		X	X	X	X	
18							X
19		X					
20							
21							X
22		X					
23	X		X	X			X
24	X						X

learning processes while creating and using computational models, thus resulting in forms of representational competence. Students were guided in processes of self-explanation as they articulated details of the implementation of their models in the form of in-code comments and the explanations of the outputs generated by their models explaining the biological phenomenon. Findings suggest that the articulation scaffolding was useful in helping students achieve working solutions for the biological problem posed to them, as evidenced by the overall level of representational competence for all teams and all projects ranging between the proficient and advanced levels of performance (over 3.5, on average). We can also conclude that students performed at the advanced level in tasks associated with creating the models and explaining the models with an overall average performance of 4.3 and over, as shown by students' understanding of all or most aspects of the code, as evidenced by their in-code commenting. It can also be concluded that students performed at a proficient level in tasks associated with creating proper outputs of their models and explaining those outputs in terms of the biological phenomenon. These findings also suggest that providing students with templates of code and test cases was an effective strategy to guide learners through the steps of implementing a solution. This is consistent with previous studies, which have identified that test cases and templates can be an effective strategy for the sense-making of complex programming topics (Denny et al., 2019; Hislop & Ellis, 2009), as well as educational evidence-based teaching practices such as the use-modify-create approach, commonly implemented in learning programming contexts (Lee & Malyn-Smith, 2020; Lee et al., 2011). Our results contribute to this body of research by elaborating on additional scaffolding, such as reflection and articulation scaffolding, which can be useful additions to help students better implement, use, and learn from computational models.

When comparing the students' level of representational competence in computational modeling, graphical representation, code interpretation, and model output explanation skill categories, we observed that the overall performance in the graphical representation scored the lowest, averaging just over 3.7, when contrasted with the other three skill categories, all of which averaged over 4.1.

It's important to highlight that, despite students receiving code scaffolds for generating plots, certain teams encountered difficulties when adapting the provided code to create precise graphs. They struggled to add just a few lines of code to ensure accuracy while maintaining crucial details, such as axis scale size, plot line colors, titles, axis labels, and plot legends. These findings suggest that students need clear guidelines to familiarize themselves with the significance of producing descriptive graphical plots. Research suggests that students may need direct instruction or scaffolding in graph-reading and graph-generating, which can result in graphical literacy skills (Witkow et al., 2022). For instance, students may benefit from tailored code scaffolding aimed at enhancing students' graphical representation skills, along with explanation guides that highlight key domain-specific variables students must elaborate upon when making argumentations about the model results. Through the reflection data, we also identified that some students experienced difficulties properly explaining the outputs of their models. This finding is consistent with prior research that suggests that students may be able to produce good plots while having little understanding of the underlying physical meaning of the plot (Nixon et al., 2016). This finding can be attributed in part to the struggles students often experience in connecting programming tasks with their real-world contexts (Magana et al., 2013; Ryoo, 2019). Thus, in future implementations of the course, we will implement and further investigate the usefulness of providing students with an argumentation framework and corresponding training that will help them align claims with evidence and supported reasoning by leveraging model results, often in the form of plots (McNeill et al., 2006).

A third outcome of the study consisted of identifying students' reported regulatory skills that were made explicit through the reflection scaffolding. Specifically, students reflected on their learnings, the challenges experienced in the process, individually and as a team, and strategies they implemented to overcome those challenges. We identified specific themes that characterized students' experienced difficulties, along with the strategies the students used to overcome those challenges. While no evident patterns were identified regarding the relationship between representational competence, challenges, and regulatory skills, it appears that teams who consistently implemented computational models at the advanced level also identified more challenges experienced in their reflections and implemented more regulatory strategies to overcome those challenges. Learning outcomes have been positively associated with metacognitive and regulatory strategies (Guo, 2022). However, since this observation is based on reflection data, there is no way to verify if these strategies were truly enacted and if they were indeed effective. Future course implementations could consider formal instruction that can promote co-regulated learning skills and the effectiveness of such instruction or scaffolding in students' improvement of their metacognitive processes (e.g., Udosen et al., 2024). Taking a co-regulated learning lens can support learners not only in their planning, monitoring, and evaluating processes but also in the central role of social transactions (Rich, 2017). Such social transactions can "emphasize shared control of learning and the role more advanced others play in scaffolding novices' metacognitive engagement" (Rich, 2017, p. 1100).

Also, while two of the major reported challenges were related to learning the concepts and skills of the course (i.e., programming models and explanations of the outputs), a third challenge was related to teamwork coordination and project management skills. For instance, from the qualitative data, it was observed that students experienced challenges with time and project management skills. Research has previously identified that time and project management skills often directly correlate to student performance in the classroom (Adams & Blair, 2019). Thus, future implementations of this course will consider providing students with additional process management scaffolding to aid student performance.

Research has consistently reported that teamwork and project management skills need to be explicitly taught, as just having students work in groups may not translate into becoming teams (Wheelan et al., 2020). Although the students were provided with process management scaffolding to provide some planning at the beginning of the semester and at the beginning of each project, more scaffolding or training was needed in order to guide them through the implementation and completion of the project. The team contracts signed at the beginning of the semester may have helped students decide on the communication medium and coordinate potential meeting times. However, more guidance is necessary to help students define internal deadlines and potential guidelines for role assignments and the distribution of work and timing within each of the project's submissions (De los Ríos-Carmenado et al., 2015).

Implications for STEM education research, teaching, and learning

The first implication of this study focuses on studying model-based learning in the context of team-based scaffolded computational modeling and simulation practices. Operationalizing model-based learning in the context of computational modeling and simulation opens up two aspects of discipline-based education research. First, we can investigate computational

model-based learning by focusing on people's knowledge and skills (Chinn et al., 2014). By looking at model-based learning this way, we can consequently design, implement, test, and further revise educational practices that can support students in the form of computational model-based learning (Seel, 2017; Simarro & Couso, 2021).

A second implication of the study contributing to the body of knowledge in discipline-based education relates to the delineation of computational modeling and simulation practices, specifically characterizing them in terms of model-based learning and their relation to representational competence and regulatory skills. Specifically, computational model-based learning occurs when learners (a) engage in the process of creating and using computational models and generating explanations of the computational models and the outputs of such models and (b) use regulatory or co-regulatory learning strategies when approaching the solution to the computational modeling and simulation projects. This finding is significant as previous research has mainly characterized model-based learning as the processes of using, creating, evaluating, or revising models (Johnson-Laird, 1995; Nersessian, 2008; Schwarz & White, 2005).

A third implication of this study is that students, as they work in teams in solving computational modeling and simulation assignments, can develop some levels of representational competence and regulatory skills. Thus, regarding teaching and learning, it is important to provide students not only with scaffolding to support students in the development of their representational competence but also learning strategies for promoting the development of self-regulated and co-regulated learning (Hadwin et al., 2018). For instance, guidelines from self-regulated learning (Zimmerman, 2000) can provide students and their teams with strategies and project management methods for engaging in processes for (a) planning, including goal setting and strategic planning, (b) performance monitoring, including deliberate self and peer instruction, time management, and metacognitive monitoring, and (c) self-reflection including self-evaluations of the work achieved and the processes involved for achieving that work. Furthermore, it is also important that students develop socially shared or co-regulated learning behaviors as they engage in problem-solving processes, share common goals, and achieve those goals through interactions with peers (Hadwin & Oshige, 2011). By doing so, students would not only improve their own self-regulated learning skills but also develop effective teamwork skills (Aggrawal et al., 2023; Tsai, 2015). If students can effectively develop self-regulated and co-regulated learning skills when scaffolding fades, the literature suggests they may be able to continue to succeed in completing their computational projects while working in teams (VanDeGrift et al., 2011).

Conclusion, limitations, and future work

The main contribution of this study is twofold. First, this study provided scaffolding guidelines in the form of sense-making, articulation, process management, and reflection, embedded in team-based computational modeling and simulation projects that promoted computational model-based learning. Second, and more importantly, for discipline-based education research, this study characterized different levels of students' engagement, more meaningful and more superficial, of computational model-based learning in the context of modeling and simulation projects operationalized as evidence for (a) analyzing the problem, planning and formulating a model, (b) implementing and evaluating the model to solve the problem, (c) using the model and interpreting the outputs of the model, and (d) reporting on the solution of the problem and reflecting on the overall process. Characterizing model-based learning in the context of computational modeling and simulation represents important contributions that showcase students' benefits and experienced challenges when engaging in model-based learning and working with others to solve computational projects. Third, the study described how computational model-based learning promoted different levels of (a) representational competence in the form of working models and accurate outputs of the models and corresponding explanations and interpretations and (b) regulatory skills that students reported they applied to overcome learning and collaboration challenges.

Despite its contributions, this study suffers from limitations associated with intervention studies with no control condition in that its design limits the generalizability of the findings to similar contexts. In this regard, the study had limited information regarding students' backgrounds, such as information regarding students' socioeconomic backgrounds or regarding non-traditional students, thus further limiting the generalizability of the findings. Also, findings associated with regulatory processes rely on self-reported data, which may or may not have been enacted by the teams or that were enacted but not reported or noticed by team members. Similarly, the reflection data may have been influenced by team dynamics, including team size, as well as the cultural or demographic backgrounds of team members. However, the rich environment of this intervention and the thorough analyses provide a unique context that contributes to our understanding of computational model-based learning. Thus, the findings have potential transferability to similar contexts. Theoretical considerations for future research involve a deeper understanding of model-based learning,

among other cognitive activities, and their applications to using computational modeling and simulation practices for solving ill-structured problems. Practical considerations for future research involve investigating the effect of promoting co-regulated learning skills and gradually removing scaffolding.

Acknowledgements
This work was supported by the U.S. National Science Foundation (NSF) under awards # 2120200 and # 2219271. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF or the U.S. Government. The authors would like to thank the editors and the reviewers for their detailed reviews, insights, and important comments that have helped improve our paper.

Author contributions
AM: conception and design of the work, interpretation of data, drafted and substantively revised the work; JA: analysis of data, interpretation of data, drafted the work; AU: analysis of data, interpretation of data, drafted the work; JL: analysis of data, interpretation of data, drafted the work; PJ: analysis of data; EP: collection of the data, interpretation of data, and substantively revised the work.

Funding
This work was supported by the U.S. National Science Foundation (NSF) under awards # 2120200, and # 2219271. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF or the U.S. Government.

Availability of data and materials
The datasets generated and/or analyzed during the current study are not publicly available due to participants' privacy and confidentiality but are available from the corresponding author upon reasonable request.

Declarations

Competing interests
The authors declare no competing interests.

Received: 20 February 2024 Accepted: 19 July 2024
Published online: 30 July 2024

References

Abdullah, S. R. S., Markom, M., & Hasan, H. A. (2013). Challenges in teaching and learning fundamentals of thermodynamics in engineering. *Journal of Engineering and Applied Science, 8*(1), 29–37.
Adams, R. V., & Blair, E. (2019). Impact of time management behaviors on undergraduate engineering students' performance. *SAGE Open, 9*(1), 2158244018824506.
Araujo, I. S., Veit, E. A., & Moreira, M. A. (2008). Physics students' performance using computational modelling activities to improve kinematics graphs interpretation. *Computers & Education, 50*(4), 1128–1140.
Arigye, J., Lyon, J. A., Magana, A. J., & Pienaar, E. (2024). The reflective modeling practitioner: Promoting self-regulation and self-confidence in computational modeling and simulation practices. *International Journal of Engineering Education, 40*(1), 179–195.
Aurigemma, J., Chandrasekharan, S., Nersessian, N. J., & Newstetter, W. (2013). Turning experiments into objects: The cognitive processes involved in the design of a lab-on-a-chip device. *Journal of Engineering Education, 102*(1), 117–140. https://doi.org/10.1002/jee.20003
Avargil, S., & Piorko, R. (2022). High school students' understanding of molecular representations in a context-based multi-model chemistry learning approach. *International Journal of Science Education, 44*(11), 1738–1766.
Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa
Chandrasekharan, S., & Nersessian, N. J. (2015). Building cognition: The construction of computational representations for scientific discovery. *Cognitive Science, 39*(8), 1727–1763.
Chandrasekharan, S., Nersessian, N. J., & Subramanian, V. (2012). Computational modeling: Is this the end of thought experiments in science? In J. Brown, M. Frappier, & L. Meyenell (Eds.), *Thought experiments in philosophy, science and the arts* (pp. 239–260). Routledge.
Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education, 95*, 202–215.
Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology, 12*(2), ep272.
Collins, A., Brown, J. S., & Newman, S. E. (2018). *Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics*. Routledge.
Cox, M. T. (2005). Metacognition in computation: A selected research review. *Artificial Intelligence, 169*(2), 104–141.
Deci, E. L., Olafsen, A. H., & Ryan, R. M. (2017). Self-determination theory in work organizations: The state of a science. *Annual Review of Organizational Psychology and Organizational Behavior, 4*, 19–43.
De los Rios-CarmenadoLopezGarcia, I. F. R. C. P. (2015). Promoting professional project management skills in engineering higher education: Project-based learning (PBL) strategy. *International Journal of Engineering Education, 31*(1), 184–198.
Develaki, M. (2017). Using computer simulations for promoting model-based reasoning: Epistemological and educational dimensions. *Science & Education, 26*(7–9), 1001–1027.
DiDonato, N. C. (2013). Effective self-and co-regulation in collaborative learning groups: An analysis of how students regulate problem solving of authentic interdisciplinary tasks. *Instructional Science, 41*, 25–47.
Edelsbrunner, P. A., Malone, S., Hofer, S. I., Küchemann, S., Kuhn, J., Schmid, R., Altmeyer, K., Brünken, R., & Lichtenberger, A. (2023). The relation of representational competence and conceptual knowledge in female and male undergraduates. *International Journal of STEM Education, 10*(1), 44.
Fennell, H. W., Lyon, J. A., Madamanchi, A., & Magana, A. J. (2020). Toward computational apprenticeship: Bringing a constructivist agenda to computational pedagogy. *Journal of Engineering Education, 109*(2), 170–176.
Granger, B., & Pérez, F. (2021). Jupyter: Thinking and storytelling with code and data. *Computing in Science & Engineering, 23*(2), 7–14. https://doi.org/10.1109/MCSE.2021.3059263
Gunawan, J. (2015). Ensuring trustworthiness in qualitative research. *Belitung Nursing Journal, 1*(1), 10–11.
Guo, L. (2022). Using metacognitive prompts to enhance self-regulated learning and learning outcomes: A meta-analysis of experimental studies in computer-based learning environments. *Journal of Computer Assisted Learning, 38*(3), 811–832.
Hadwin, A., Järvelä, S., & Miller, M. (2018). Self-regulation, co-regulation, and shared regulation in collaborative learning environments. In D. H. Schunk & J. A. Greene (Eds.), *Handbook of self-regulation of learning and performance* (pp. 83–106). Routledge/Taylor & Francis Group.
Hadwin, A., & Oshige, M. (2011). Self-regulation, coregulation, and socially shared regulation: Exploring perspectives of social in self-regulated learning theory. *Teachers College Record, 113*(2), 240–264.
Hallström, J., Norström, P., & Schönborn, K. J. (2023). Authentic STEM education through modelling: An international Delphi study. *International Journal of STEM Education, 10*(1), 62.
Hallström, J., & Schönborn, K. J. (2019). Models and modelling for authentic STEM education: Reinforcing the argument. *International Journal of STEM Education, 6*(1), 1–10.
Hawe, E., Lightfoot, U., & Dixon, H. (2019). First-year students working with exemplars: Promoting self-efficacy, self-monitoring and self-regulation. *Journal of Further and Higher Education, 43*(1), 30–44.
Helikar, T., Cutucache, C. E., Dahlquist, L. M., Herek, T. A., Larson, J. J., & Rogers, J. A. (2015). Integrating interactive computational modeling in biology curricula. *PLOS Computational Biology, 11*(3), e1004131.
Hsieh, H.-F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research, 15*(9), 1277–1288.

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), 1–16.

Ifenthaler, D., & Seel, N. M. (2013). Model-Based Reasoning. *Computers & Education, 64*, 131–142.

Izsak, A. (2004). Students' coordination of knowledge when learning to model physical situations. *Cognition and Instruction, 22*(1), 81–128.

Jaiswal, A., Lyon, J. A., Zhang, Y., & Magana, A. J. (2021). Supporting student reflective practices through modelling-based learning assignments. *European Journal of Engineering Education, 46*(6), 987–1006.

Johnson-Laird, P. N. (1995). Mental models, deductive reasoning, and the brain. *The Cognitive Neurosciences, 65*, 999–1008.

Kim, M. C., & Hannafin, M. J. (2011). Scaffolding problem solving in technology-enhanced learning environments (TELEs): Bridging research and theory with practice. *Computers & Education, 56*(2), 403–417.

Lee, I., & Malyn-Smith, J. (2020). Computational thinking integration patterns along the framework defining computational thinking from a disciplinary perspective. *Journal of Science Education and Technology, 29*, 9–18.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads, 2*(1), 32–37.

Liu, H., Spector, J. M., & Ikle, M. (2018). Computer technologies for model-based collaborative learning: A research-based approach with initial findings. *Computer Applications in Engineering Education, 26*(5), 1383–1392.

Löhner, S., van Joolingen, W. R., Savelsbergh, E. R., & van Hout-Wolters, B. (2005). Students' reasoning during modeling in an inquiry learning environment. *Computers in Human Behavior, 21*(3), 441–461.

Lyon, J. A., & Magana, A. J. (2021). The use of engineering model-building activities to elicit computational thinking: A design-based research study. *Journal of Engineering Education, 110*(1), 184–206. https://doi.org/10.1002/jee.20372

Lyon, J. A., Magana, A. J., & Streveler, R. A. (2022). Characterizing computational thinking in the context of model-planning activities. *Modelling, 3*(3), 344–358.

Magana, A. J. (2024). *Teaching and learning in STEM with computation, modeling, and simulation practices: A guide for practitioners and researchers*. Purdue University Press.

Magana, A. J., Falk, L. M., & Reese, J. M. (2013). Introducing discipline-based computing in undergraduate engineering education. *ACM Transactions on Computing Education, 13*(4), 1–22. https://doi.org/10.1145/2534971

Magana, A. J., Falk, L. M., Vieira, C., Reese, M. J., Jr., Alabi, O., & Patinet, S. (2017). Affordances and challenges of computational tools for supporting modeling and simulation practices. *Computer Applications in Engineering Education, 25*(3), 352–375. https://doi.org/10.1002/cae.21804

Magana, A. J., Fennell, H. W., Vieira, C., & Falk, M. L. (2019). Characterizing the interplay of cognitive and metacognitive knowledge in computational modeling and simulation practices. *Journal of Engineering Education, 108*(2), 276–303.

Magana, A. J., & Mathur, J. I. (2012). Motivation, awareness, and perceptions of computational science. *Computing in Science & Engineering, 14*(1), 74–79.

Magana, A. J., Vieira, C., Fennell, H. W., Roy, A., & Falk, M. L. (2020). Undergraduate engineering students' types and quality of knowledge used in synthetic modeling. *Cognition and Instruction*. https://doi.org/10.1080/07370008.2020.1792912

Malone, K. L., Schunn, C. D., & Schuchardt, A. M. (2018). Improving conceptual understanding and representation skills through excel-based modeling. *Journal of Science Education and Technology, 27*(1), 30–44.

McNeill, K. L., Lizotte, D. J., Krajcik, J., & Marx, R. W. (2006). Supporting students' construction of scientific explanations by fading scaffolds in instructional materials. *The Journal of the Learning Sciences, 15*(2), 153–191.

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education, 62*(2), 77–90.

Nersessian, N. J. (2002a). The cognitive basis of model-based reasoning in science. In P. Carruthers, S. Stich, & M. Siegal (Eds.), *The cognitive basis of science* (pp. 133–153). Cambridge University Press. https://doi.org/10.1017/CBO9780511613517.008

Nersessian, N. J. (2002b). The cognitive basis of model-based reasoning in science. In P. Carruthers, S. Stich, & M. Siegal (Eds.), *The cognitive basis of science* (pp. 133–153). Cambridge University Press.

Nersessian, N. J. (2009). How do engineering scientists think? Model-based simulation in biomedical engineering laboratories. *Topics in Cognitive Science, 1*(4), 730–757.

Nixon, R. S., Godfrey, T., Mayhew, N. T., & Wiegert, C. C. (2016). Undergraduate student construction and interpretation of graphs in physics lab activities. *Physical Review Physics Education Research, 12*(1), 010104.

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.

Ortega-Alvarez, J. D., Sanchez, W., & Magana, A. J. (2018). Exploring undergraduate students' computational modeling abilities and conceptual understanding of electric circuits. *IEEE Transactions on Education, 61*(3), 204–213.

Oyelere, S. S., Olaleye, S. A., Balogun, O. S., & Tomczyk, Ł. (2021). Do teamwork experience and self-regulated learning determine the performance of students in an online educational technology course? *Education and Information Technologies, 26*(5), 5311–5335.

Penner, D. E. (2000). Cognition, computers, and synthetic science: Building knowledge and meaning through modeling. *Review of Research in Education, 25*, 1–36.

Peters, E. E., & Kitsantas, A. (2010). Self-regulation of student epistemic thinking in science: The role of metacognitive prompts. *Educational Psychology, 30*(1), 27–52.

Psycharis, S., Botsari, E., Mantas, P., & Loukeris, D. (2014). The impact of the computational inquiry based experiment on metacognitive experiences, modelling indicators and learning performance. *Computers & Education, 72*, 90–99.

Quillin, K., & Thomas, S. (2015). Drawing-to-learn: a framework for using drawings to promote model-based reasoning in biology. *Cbelife Sciences Education, 14*(1), es2.

Quintana, C., Reiser, B. J., Davis, E. A., Krajcik, J., Fretz, E., Duncan, R. G., Kyza, E., Edelson, D., & Soloway, E. (2004). A scaffolding design framework for software to support science inquiry. *The Journal of the Learning Sciences, 13*(3), 337–386.

Rich, J. V. (2017). Proposing a model of co-regulated learning for graduate medical education. *Academic Medicine, 92*(8), 1100–1104.

Rittle-Johnson, B., Siegler, R. S., & Alibali, M. W. (2001). Developing conceptual understanding and procedural skill in mathematics: An iterative process. *Journal of Educational Psychology, 93*(2), 346.

Ryoo, J. J. (2019). Pedagogy that supports computer science for all. *ACM Transactions on Computing Education (TOCE), 19*(4), 1–23.

Sanchez-Pena, M., Vieira, C., & Magana, A. J. (2022). Data science knowledge lintegration: Affordances of a computational cognitive apprenticeship on student conceptual understanding. *Computer Applications in Engineering Education*. https://doi.org/10.1002/cae.22580

Schwarz, C., & White, B. Y. (2005). Metamodeling knowledge: Developing students' understanding of scientific modeling. *Cognition and Instruction, 23*(2), 165–205.

Seel, N. M. (2017). Model-based learning: A synthesis of theory and research. *Educational Technology Research and Development, 65*, 931–966.

Şen, Ş, Yılmaz, A., & Geban, Ö. (2015). The effects of process oriented guided inquiry learning environment on students' self-regulated learning skills. *Problems of Education in the 21st Century, 66*, 54.

Shiflet, A. B., & Shiflet, G. W. (2014). *Introduction to computational science: Modeling and simulation for the sciences*. Princeton University Press.

Simarro, C., & Couso, D. (2021). Engineering practices as a framework for STEM education: A proposal based on epistemic nuances. *International Journal of STEM Education, 8*(1), 1–12.

Sins, P. H., Savelsbergh, E. R., & van Joolingen, W. R. (2005). The difficult process of scientific modelling: An analysis of novices' reasoning during computer-based modelling. *International Journal of Science Education, 27*(14), 1695–1721. https://doi.org/10.1080/09500690500206408

Stahl, N. A., & King, J. R. (2020). Expanding approaches for research: Understanding and using trustworthiness in qualitative research. *Journal of Developmental Education, 44*(1), 26–28.

Stieff, M., Scopelitis, S., Lira, M. E., & Desutter, D. (2016). Improving representational competence with concrete models. *Science Education, 100*(2), 344–363.

Stull, A. T., Gainer, M., Padalkar, S., & Hegarty, M. (2016). Promoting representational competence with molecular models in organic chemistry. *Journal of Chemical Education, 93*(6), 994–1001.
Stull, A. T., & Hegarty, M. (2016). Model manipulation and learning: Fostering representational competence with virtual and concrete models. *Journal of Educational Psychology, 108*(4), 509.
Sun, Y., Newstetter, W., & Nersessian, N. J. (2006). *Promoting model-based reasoning in problem-based learning*. Trabajo presentado en la reunión anual de la Cognitive Science Society.
Tee, K. N., Leong, K. E., & Abdul Rahim, S. S. (2021). A self-regulation model of mathematics achievement for eleventh-grade students. *International Journal of Science and Mathematics Education, 19*(3), 619–637.
Tsai, C.-W. (2015). The effect of online co-regulated learning in the implementation of team-based learning on improving students' involvement. *Higher Education Research & Development, 34*(6), 1270–1280.
Turner, P. R., Petzold, L., Shiflet, A., Vakalis, I., Jordan, K., & John, S. S. (2011). Undergraduate computational science and engineering education. *SIAM Review, 53*(3), 561–574.
Vaismoradi, M., Jones, J., Turunen, H., & Snelgrove, S. (2016). Theme development in qualitative content analysis and thematic analysis. *Journal of Nursing Education and Practice, 6*(5), 100–110.
Vieira, C., Magana, A. J., Falk, M. L., & Garcia, R. E. (2017). Writing in-code comments to self-explain in computational science and engineering education. *ACM Transactions on Computing Education (TOCE), 17*(4), 17:01-17:21. https://doi.org/10.1145/3058751
Vieira, C., Magana, A. J., Garcia, R. E., Jana, A., & Krafcik, M. (2018). Integrating computational science tools into a thermodynamics course. *Journal of Science Education and Technology, 27*(1), 1–12. https://doi.org/10.1007/s10956-017-9726-9
Vieira, C., Magana, A. J., Roy, A., Falk, L. M., & Reese, J. M. (2016a). Exploring undergraduate students' computational literacy in the context of problem solving. *Computers in Education Journal., 7*(1), 100–112.
Vivek, R., & Nanthagopan, Y. (2021). Review and comparison of multi-method and mixed method application in research studies. *European Journal of Management Issues, 29*(4), 200–208.
Vygotsky, L. S., & Cole, M. (1978). *Mind in society: Development of higher psychological processes*. Harvard University Press.
Watts, F. M., & Finkenstaedt-Quinn, S. A. (2021). The current state of methods for establishing reliability in qualitative chemistry education research articles. *Chemistry Education Research and Practice, 22*(3), 565–578.
Wheelan, S. A., Åkerlund, M., & Jacobsson, C. (2020). *Creating effective teams: A guide for members and leaders*. Sage Publications.
Witkow, M. R., Spiekerman, A., Ford-Roshon, C., Hershman, T., & Stevens, C. (2022). An experimental study of the effects of scaffolding and context in training graphical literacy among 2-and 4-year college students. *Scholarship of Teaching and Learning in Psychology*. https://doi.org/10.1037/stl0000344
Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry, 17*(2), 89–100.
Zhang, B., Liu, X., & Krajcik, J. S. (2006). Expert models and modeling processes associated with a computer-modeling tool. *Science Education, 90*(4), 579–604.
Zimmerman, B. J. (2000). Attaining self-regulation: A social cognitive perspective. In P. R. Pintrich & M. Zeidner (Eds.), *Handbook of self-regulation* (pp. 13–39). Elsevier.
Zimmerman, B. J., & Paulsen, A. S. (1995). Self-monitoring during collegiate studying: An invaluable tool for academic self-regulation. *New Directions for Teaching and Learning, 1995*(63), 13–27.
Zwickl, B. M., Hu, D., Finkelstein, N., & Lewandowski, H. (2015). Model-based reasoning in the physics laboratory: Framework and initial results. *Physical Review Special Topics-Physics Education Research, 11*(2), 020113.
Aggrawal, S., Cristancho, J. A., Patel, D. A., & Magana, A. J. (2023). *Cooperative Learning and Co-Regulation: Exploring Students' Teamwork Strategies in Higher Education.* Paper presented at the 2023 IEEE Frontiers in Education Conference (FIE).
Chinn, C. A., Rinehart, R. W., & Buckland, L. A. (2014). Epistemic cognition and evaluating information: Applying the AIR model of epistemic cognition. *Processing inaccurate information: Theoretical and applied perspectives from cognitive science and the educational sciences*, 425–453.
Clement, J. J., & Rea-Ramirez, M. A. (2008). Model based learning and instruction in science. *Model based learning and instruction in science*, 1–9.
Denny, P., Prather, J., Becker, B. A., Albrecht, Z., Loksa, D., & Pettit, R. (2019). *A closer look at metacognitive scaffolding: Solving test cases before programming.* Paper presented at the Proceedings of the 19th Koli Calling international conference on computing education research, New York, NY.
Garrison, D. R., & Akyol, Z. (2015). Thinking collaboratively in educational environments: Shared metacognition and co-regulation in communities of inquiry *Educational developments, practices and effectiveness: Global perspectives and contexts* (pp. 39–52): Springer.
Hislop, G. W., & Ellis, H. J. (2009). *Using scaffolding to improve written communication of software engineering students.* Paper presented at the 2009 Sixth International Conference on Information Technology: New Generations, Las Vegas, NV.
Lehrer, R., & Schauble, L. (2003). Origins and evolution of model-based reasoning in mathematics and science *Beyond Constructivism* (pp. 59–70). Routledge.
Lehrer, R., & Schauble, L. (2006). Cultivating model-based reasoning in science education. *Cambridge handbook of the learning sciences*, 371–388.
Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., et al. (2018). *Introductory programming: a systematic literature review.* Paper presented at the Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education.
Madamanchi, A., Cardella, M. E., Glazier, J. A., & Umulis, D. M. (2018). *Factors mediating learning and application of computational modeling by life scientists.* Paper presented at the 2018 IEEE Frontiers in Education Conference (FIE).
Mashood, K., Khosla, K., Prasad, A., Sasidevan, V., CH, M. A., Jose, C., & Chandrasekharan, S. (2022). Participatory approach to introduce computational modeling at the undergraduate level, extending existing curricula and practices: Augmenting derivations. *Physical Review Physics Education Research, 18*(2), 020136.
Mayer, R. E. (1985). Learning in complex domains: A cognitive analysis of computer programming *Psychology of learning and motivation* (Vol. 19, pp. 89–130): Elsevier.
Munahefi, D., & Waluya, S. (2018). *Analysis of creative mathematic thinking ability in problem based learning model based on self-regulation learning.* Paper presented at the Journal of Physics: Conference Series.
Nersessian, N. J. (2008). Model-based reasoning in scientific practice *Teaching scientific inquiry* (pp. 57–79): Brill sense.
Nersessian, N. J., & Patton, C. (2009). Model-based reasoning in interdisciplinary engineering *Philosophy of technology and engineering sciences* (pp. 727–757): Elsevier.
Sabourin, J., Mott, B., & Lester, J. (2013). *Discovering behavior patterns of self-regulated learners in an inquiry-based learning environment.* Paper presented at the Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9–13, 2013. Proceedings 16.
Santos, S. C., Tedesco, P. A., Borba, M., & Brito, M. (2020). *Innovative approaches in teaching programming: A systematic literature review.* Paper presented at the Proceedings of the 12th International Conference on Computer Supported Education.
Shaikh, U. A., Magana, A. J., Vieira, C., & Garcia, R. E. (2015). *An exploratory study of the role of modeling and simulation in supporting or hindering engineering students' problem solving skills.* Paper presented at the Proceedings of the 2015 ASEE Annual Conference and Exposition. Seattle, Washington.
Udosen, A. N., Magana, A. J., & Pienaar, E. (2024). *Characterizing teamwork dynamics and computational model-based reasoning in biomedical engineering projects.* Paper presented at the 2024 ASEE Annual Conference & Exposition, Portland, Oregon.
van Borkulo, S., van Joolingen, W. R., Savelsbergh, E. R., & de Jong, T. (2008). Model-based approaches to learning. In P. Blumschein, J. Stroebel, W. Hung, & D. Jonassen (Eds.): Sense Publishers.
VanDeGrift, T., Caruso, T., Hill, N., & Simon, B. (2011). *Experience report: Getting novice programmers to think about improving their software development process.* Paper presented at the Proceedings of the 42nd ACM technical symposium on Computer science education.
Vieira, C., Magana, A. J., Roy, A., Falk, L. M., & Reese, J. M. (2015). *Exploring undergraduate students' computational literacy in the context of problem solving.* Paper presented at the Proceedings of the 122th ASEE Annual Conference and Exposition, Seattle, Washington.

Vieira, C., Roy, A., Magana, A. J., Falk, M. L., & Reese, M. J. (2016). *In-code comments as a self-explanation strategy for computational science education.* Paper presented at the 2016 ASEE Annual Conference & Exposition.

Wang, A. Y., Mittal, A., Brooks, C., & Oney, S. (2019). How data scientists use computational notebooks for real-time collaboration. *Proceedings of the ACM on Human-Computer Interaction, 3*(CSCW), 1–30.

Publisher's Note