Talking with Cody: How Middle School Students Enact AI as a Tool, Conversational Partner, and Tutee in Mathematics Programming Activities

Anders I. Mørch[1*], Marianne Eidsaune Andersen[1] and Henrik Tjønn[1]

[1] *Department of Education, University of Oslo, Helga Eng bldg., Sem Sælands vei 7, 0371 Oslo, Norway*

Abstract

Programming became mandatory in the Norwegian mathematics curriculum (K–12) in the 2020 reform, to support exploration of mathematical properties, algorithmic thinking, and working with data. This study examines how GenAI is integrated into such programming-based mathematics activities, and how this shapes the role of AI in classroom interaction. We analyse video data from a 9th grade middle school class working on geometric figures (right-angled triangles and trapezoids) with Python turtle. The students first solved tasks without AI and then used the school-provided chatbot Cody to generate or explain code. Using thematic analysis informed by a conceptual framework of AI scaffolding strategies, articulated at both latent and semantic levels in the students' utterances, we identify three enacted roles for Cody: (1) *tool*, when students copy, run, and sometimes edit Cody's code in their own environment; (2) *conversational partner*, when they negotiate prompts, and iteratively refine and reflect on Cody's responses; and (3) *tutee*, when they describe and supply missing geometric constraints. The unanticipated tutee role suggests that conversational AI in a constructionist mathematics-programming classroom not only tutors students, but also is customized and corrected by them as a light form of end-user development.

Keywords

AI in education, AI scaffolding, AI-as-tutee, end-user development, programming as empirical inquiry

1. Introduction

In research on meta-design and End-User Development (EUD), learners are increasingly seen as co-designers who shape socio-technical environments through their own adaptations, including programming activities [1]. Generative artificial intelligence (GenAI) is one domain where EUD is now being applied [2]. In our case, we treat the AI assistant and the surrounding classroom practices as a meta-designed environment: municipal actors and teachers configure a school-provided AI chatbot and the initial task structure [3], and students later iteratively correct its suggestions during their geometry work. This learner-driven adaptation shows how meta-design also emerges as a student practice, as they reflectively adjust prompts and tasks constraints to reshape their own learning environment from within.

Our study examines how Cody, a locally adapted version of ChatGPT 4.o for learning elementary programming, can act as an AI assistant for middle school students working on geometry-related programming tasks in mathematics.

We are inspired by Schön's [4] notion of designing as a "reflective conversation with the materials of a design situation," extended here to include code, its outputs, and AI responses as materials of empirical inquiry, and by recent studies that treats computational thinking, as learning to program through empirical inquiry [5, 6, 7], where students iteratively "converse" with their code and its runtime environment, asking what the program does, observing its behaviour, and revising their ideas in response to outputs and errors [5, 4].

· *Proceedings of the 10th International Workshop on Cultures of Participation in the Digital Age (CoPDA 2026): Exploring the Relationship between EUD, AI-Assisted Development, and Meta Design, June 2026, Venice, Italy.*

* Corresponding author.

andersm@uio.no (A. Mørch); marianne.eidsaune.andersen@gmail.com (M.E. Andersen); henriktj@uio.no (H. Tjønn)

0000-0002-1470-5234 (A. Mørch); 0009-0007-2035-4390 (H. Tjønn)

Building on this inquiry perspective on learning to program, set within a constructionist programming learning environment [8], we analyse classroom episodes in which students negotiate mathematical meaning with Cody and with each other. We therefore address the following research question: *How is the school-adapted chatbot Cody enacted as an AI assistant in middle school mathematics programming activities?*

To address this question, we first provide a brief review of related work and introduce our analytical framework, drawing on programming-as-empirical-inquiry and conversation perspectives. We then present our data collection and empirical methods, followed by our results, where we use selected data excerpts to illustrate and analyse different enactments of Cody. Finally, we discuss our findings in relation to the existing literature and suggest directions for future research.

2. Related work and analytical framework

2.1. AI in education

Language use is fundamental to teaching, learning, and knowledge development [9]. The introduction of AI in education presents both challenges and opportunities, as these systems can automate educational tasks for both students and teachers [10]. However, simply making tasks easier or more automated does not necessarily support learning, which depends on students engaging in substantive cognitive and collaborative work with developing ideas rather than offloading it to AI [11, 12]. Kasneci and colleagues [13] discuss the educational potential of large language models (LLMs), suggesting that using GenAI systems can enhance cognitive abilities and technological literacy, but they also stress the need to align AI use with existing teaching methods, classroom activities, and writing practices. When LLMs such as ChatGPT are integrated into curricula, this entails incorporating new pedagogical practices, such as prompt creation and automated feedback, into lessons [14].

A recent study comparing two AI systems with respect to how their automated feedback aligns with educational practice distinguished between AI as teaching assistant and AI as learning partner [3]. In the former, the AI agent takes on a teacher-like role by providing feedback that extends the teacher's capacity to evaluate student writing and guide learning, reinforcing a traditional model of teacher authority. In the latter, the AI acts as a learner partner, inviting students to interact with the chatbot in ways that foster collaboration, reflective thinking, and student initiative.

2.2. Programming in mathematics education

Constructionism developed by Papert, Resnick, and colleagues [8, 15] emphasizes the active role that students can take in constructing their own learning through hands-on activity with physical or visual objects. A constructionist approach to geometry learning entails reconstructing the shape of a geometric figure by drawing it with an on-screen turtle, tracing its angles and sides as correct as possible [16, 8]. Key aspects of such pedagogy are design failure and sense making, which on our case is supported by teachers' instruction and scaffolding for task completion [17, 18], and AI-assisted scaffolding can complement teachers' scaffolding across a wide range of student tasks [19, 20].

Misfeldt and colleagues [21] show how the relationship between mathematics and programming is realized differently in Denmark, England, and Sweden. In one case, programming is treated mainly as a separate technical skill with few explicit links to mathematical ideas. In another, programming serves as a tool for expressing algebraic concepts, though coding challenges can overshadow mathematical goals. In the third, mathematical and programming structures (e.g., statistics, modelling, problem solving) are explicitly juxtaposed, so students encounter programming as an integrated way of working with mathematics. Across the cases, the authors illustrate how curricular intentions and task design shape whether programming appears as separate, tool-like, or integrated with mathematics.

Fojcik [22] studied Norwegian mathematics teachers' conceptions of programming in mathematics education, independent of programming language choice (block-based or text-based) and found that teachers are generally positive, but also severely underprepared and lacking institutional support, pointing to clear needs for professional development, shared language, and support structures for integrating programming into mathematics.

In summary, previous work on AI in education has largely framed LLM-based systems as tutors in the intelligent tutoring systems (ITS) tradition. At the same time, research on constructionist mathematics environments has examined students' sense-making and design activities with code and turtle graphics as tools for exploring geometry, but without an AI agent in the loop. Our study differs by analysing how a school-adapted chatbot is practically used in ordinary classroom programming, showing how students enact Cody not only as a tool or tutor-like conversational partner, but also in a way where the interaction itself becomes an object of investigation in their geometry work (e.g., iteratively testing and revising Cody's suggestions in a design conversation that resembles EUD).

2.3. Analytic concepts for AI assistant strategies

Work on ITS has long shown that AI can approximate the effectiveness of human tutoring for stepwise guidance and feedback in well-structured domains, but also that such systems tend to narrow learning activity into highly scripted interactions [10]. More recent GenAI-oriented work broadens this view by distinguishing different types of AI assistance, including more open-ended "partners" that co-construct solutions with learners [23], and by emphasising hybrid human–AI constellations [24]. These typologies, however, still focus mainly on how AI delivers instruction or optimizes performance.

In contrast, we build on Fischer's [25] three-level typology of learning strategies in guided, open-ended socio-technical and social media design environments. This perspective aligns with our view of programming as empirical inquiry, where error messages, system feedback, and observable runtime behavior are resources for sense-making through reflective conversation rather than merely obstacles to be removed [4, 5]. Fischer's [25] three learning strategies (Fix-it, Reflect, Tutorial) were adapted to conceptualize the roles of AI assistance in our 9th grade mathematics classroom, using Tool, Conversation partner, and Tutee as follows:

1. *Tool (Fix-it).* Cody is treated as a time-saving instrument that "fixes" code or produces ready-made solutions so that students can keep working, with limited demands on their own conceptual understanding.
2. *Conversational partner (Reflect).* Cody is used to extend students' ongoing empirical inquiry: they predict what a program will do, ask Cody for an explanation, compare Cody's answer with their own, and reflect on the differences to refine their prompts-based reasoning about code and geometry properties.
3. *Tutee (Tutorial).* Cody becomes an object of inquiry that the learner actively "teaches" by correcting its mistakes in drawing geometric shapes, adding missing constraints, and refining its output and explanations through dialogue.

The third learning strategy is where our data diverges from Fischer's original expectation of a tutorial role: instead of Cody tutoring the students, students sometimes position Cody as a tutee that they correct and constrain.

3. Methods

3.1. Task, participants, data collection, and data sampling

Data were collected over two days in two middle school mathematics classes (8th and 9th grade) that were working with programming. In both classes, the classroom was arranged so that students

worked in small groups at clusters of desks with one laptop, and they were encouraged to use Cody as support on selected tasks (Figure 1).

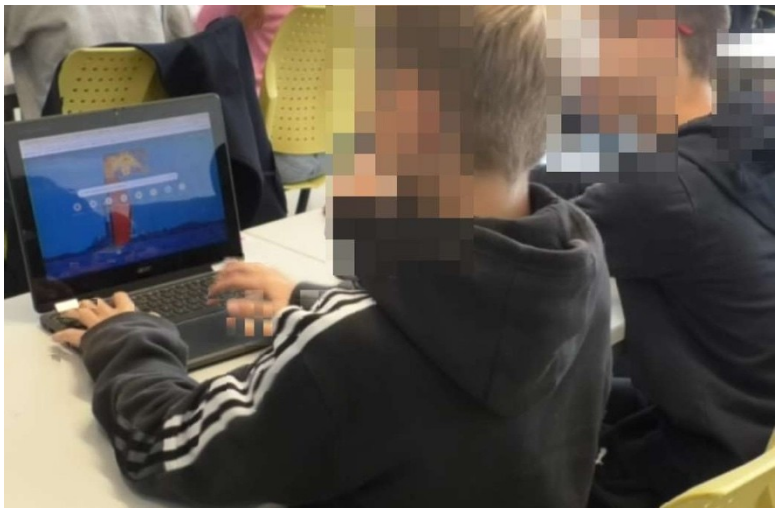

Figure 1: Classroom setting with the observed group of boys (left and right).

Before data collection, we held an online planning meeting with both teachers to agree on the lesson structure and when Cody should be introduced. On day 1, students used Python turtle to program different figures (a right-angled triangle, a right-angled trapezoid, a regular polygon using loops, and a self-designed polygon). They were asked to first develop a solution themselves, then ask Cody for help, and finally compare Cody's solution with their own. The teachers encouraged students to ask Cody for help and to explain its suggestions, but they did not explicitly frame Cody as a tutee and were not told to correct Cody's suggestions. On day 2, students interpreted a given Python turtle program, explained what happened when it was run, and then modified it to draw a multi-faceted polygon.

Data were collected by two master's students, one PhD student, and one researcher using video-based observation and interviews. Three groups of students were video recorded each day, for approximately two hours per group, yielding about 12 hours of primary video data. In addition, we conducted ten semi-structured interviews afterwards, including interviews with both teachers. The video recordings covered the whole lesson, from the teacher's introduction to the end of the task. The interviews were used as supplementary data and combined with the video material for methodological triangulation [26]. For reasons of space, we focus primarily on the video data in this paper.

We made two strategic sampling decisions for the video data. First, following teacher recommendations, we selected one 9th-grade all-boy group and one 8th-grade all-girl group, as these students were expected to be communicative and engaged. When the girl group later withdrew, we focused on the boy group because it provided a continuous dataset across both days, enabling repeated, fine-grained analysis of interactional episodes [27]. This also means our primary data come from an all-boy group.

The study was approved by the Norwegian data protection authorities, which require that participants give free and informed consent before inclusion. Students were informed about the project and received a consent form that had to be signed by their parents. Most students participated; those without consent attended the mathematics lesson in a different room with another teacher.

3.2. Transcription and thematic analysis

All interviews and video excerpts were automatically transcribed using the university's transcription service (based on Whisper, OpenAI). The authors then checked and corrected the transcripts against

the videos to ensure accuracy and consistent representation of participants' utterances, as recommended in qualitative methods literature on thematic analysis [28].

We used thematic analysis to identify and analyse patterns in the data [28]. Our coding combined inductive (bottom-up, data-driven) and deductive (top-down, theory-informed) approaches. In an initial inductive round, we developed three broad themes: students collaborate when interacting with AI, AI as support, and students' understanding of mathematics and programming. For the workshop, we focused on subcategories of "AI as support" for a more in-depth (deductive) analysis, using our adaptation of Fischer's [25] three AI assistance strategies, Fix-it, Reflect, and Tutorial, operationalized in our analysis as Tool, Conversation partner, and Tutee. Figure 2 shows parts of the coding protocol. In both rounds of coding, we based our thematic analysis of the students' utterances on two levels: semantic (explicit meaning) and latent (implicit meaning) [28, p. 35]. In the data presentations below, researchers' comments are parenthesized.

Utterance example	Sub-theme	Main theme
"We need a little help with making a right-angled triangle with Python Turtle"(two boys)	**Tool**. Cody gives a direct solution so the learner can keep working without delay, but it requires little understanding.	**AI as support**
"So it's a special kind of pentagon. Ask Cody: what kind of pentagon has equal angles of 72 degrees?"		
"Fun… okay, let's see if this is right then (copies Cody's answer). Let's see, this is the one with the side that was much longer."(boy left)	**Conversational partner**. Cody extends students' ongoing empirical inquiry: they verbalize what they think, predict what the code will do, and ask Cody for explanations.	
"Okay, so we're going to figure out what that figure will be if we replace the a's with 70 and the b's with 5, right?" (boy in the middle)		
"He didn't even do the height, and his top angle was facing the wrong way."(boy middle)	**Tutee**. Students verbalize how Cody becomes an object of inquiry that the learner actively "teaches".	
"Now I feel like we're the ones teaching Cody here…" (boy left)		

Figure 2: Part of the coding protocol developed during thematic analysis.

4. Findings (data and analysis)

The students' inquiry process over the two days was partly shaped by the teacher's task design and guidance, and partly by the students themselves as they gradually learned to interact with Cody, which included (1) using it as a tool, (2) improving joint understanding, and (3) modifying Cody's behaviour when its suggestions did not match their own programs or the intended geometric figures. We use the modified version of Fischer's framework to organize our analysis.

4.1. Cody as a tool

Cody knows a lot about programming and could answer all questions but sometimes they would not be understood, or the students would interpret it wrongly. A common pattern is to use Cody to generate code, copy the code Cody and paste into Tricket, the code editor, before running it to see the result (a geometric figure visible on the screen).

4.1.1. Extract 1 (19:50-20:40, day1) - code-producing tool

This data extract is part of a prompt issued to Cody read out loud and occurs about 20 minutes into the class on the first day. The three boys have created turtle code to draw a right-angled triangle and now they ask Cody to do the same:

Boy left: *"We're a group of two … we need a bit of help … making a right-angled triangle with Python Turtle … the base is 150 and the height is 50."*

In Extract 1, the boys are not yet "doing mathematics" or "doing Turtle," but mainly working on how to talk to Cody by prompting. They are collaboratively constructing a prompt and use Cody to get it done: "*we need a bit of help … making a right-angled triangle*," rather than "help us reflect on or debug our own code." This shows that the students are treating Cody primarily as a code-producing tool while still doing important framing and representational work in how they formulate their prompt in terms of basic vocabulary in programming and geometry. However, mathematical reasoning becomes secondary.

4.1.2. Extract 2 (34:18-34:52, day 2) – look-up (one-shot) query

The teacher had instructed the students to use Cody for different purposes, and here the suggestion is to use Cody much like a search engine. This use is proposed by the teacher because it invites "talking geometry" by stressing that naming the observed shapes will trigger conceptual understanding:

Teacher: "You're just drawing lines in the figure, right? But write it in words when it comes to this one, the pentagon. It's correct that it's a five-sided figure, but what kind of pentagon? Think about the angles."

Boy left: "72 degrees."

Teacher: "72 degrees for all the angles."

Boy middle: "Yeah."

Teacher: "So it's a special kind of pentagon. Ask Cody: what kind of pentagon has equal angles of

72 degrees?"

Boy left: "Okay."

In Extract 2, Cody is positioned as a *lookup tool*: ("Ask Cody: what kind of pentagon has equal angles of 72 degrees") rather than to reason about its properties. The conceptual work on geometry (that a regular pentagon has interior angles of 72 degrees) is still strongly guided orally by the teacher, while the students' role is reduced to confirming a label that can then be "pasted in" to the written answer. This one-shot, non-iterative use contrasts with later episodes where Cody's outputs and errors become objects of empirical inquiry.

4.2. Cody as conversational partner

In this mode the interaction with Cody is visibly dialogic: greetings, contextual self-presentations, iterative back-and-forth (turn taking), complaining, joking. Typical interactions were how to react to output students do not understand and how to fix code received from Cody when its turtle behavior does not compare with their own prior work. During these interactions, Cody sometimes helps to narrow the gap between code and geometry, and sometimes widens it. Below, we present an instance of the latter when they go into dialog with output that they find "weird shapes".

4.2.1. Extract 3 (22:13 → 23:26, day 1) – iterative, playful talk (widening the gap)

(They get a new piece of code from Cody and copy it.)

Boy left: But this here is just … (runs the code)

Boy right: It's pretty much the same, isn't it? (the figure is like before)

Boy left (pointing at the screen): The base is 150 …

Boy left: …what are we even supposed to call this shape, like, and you …

Boy right: You're making a masterpiece.

Boy left (typing to Cody): You're making …

Boy right: .. weird shapes.

Boy left (finishes typing): You're making weird shapes.

Boy right: Just say "can we…" we could try making a right-angled triangle.
Boy left: There are probably people who don't even know what a right-angled triangle is, so you look it up here and then you're like, "Oh yeah, that's a right-angled triangle."

In Extract 3, Cody keeps producing new versions of code that do not yield the intended right-angled triangle, thereby widening rather than narrowing the gap between code and geometry. The boys engage in iterative copy–paste–run loops and switch back and forth between Cody and Trinket, but without systematically relating the "weird shapes" on the screen to specific commands or angle/length choices in the code, so little of the geometric reasoning emphasized elsewhere in the lesson is taken up here. Their activity still constitutes a form of empirical inquiry, where they explore and respond to the behaviour of code and AI output as a "reflective conversation" with the runtime display on the screen (geometrical shapes). However, because the focus of that reflection is on getting something to "work" rather than on why the geometry is wrong, Cody's persistent errors here lead to characterizing the output as "weird shapes" rather than to deeper geometric understanding. In this episode, Cody's code is syntactically correct but misaligned with the intended geometry; the main difficulty lies in how the students interpret and specify constraints, not in model hallucination.

4.3. Cody as tutee

In the last category, students move to a higher level of abstraction by interacting with Cody as an object of inquiry; they compare Cody's output with their own code and with the task, so that Cody's suggestions and explanations themselves become objects of empirical inquiry. Still working within a conversational frame, they approach the geometry domain by invoking geometric properties (e.g., side lengths and angles) to diagnose and "help" Cody, aligning with a inquiry perspective on programming but with a new twist. We illustrate in the next extract:

4.3.1. *Extract 4 (23:41→ 26:26, day 1) – critique and modify (narrowing the gap)*

(Boy left copies Cody's code, goes to Python, deletes what was there and pastes in the new one.)
Boy left: "Yeah, then we'll just delete all of that again. Why is he making it so complicated?"
Boy right: "Yeah, I don't get it, what is he even doing?" (They go from Tricket back to Cody.)
Boy left, typing to Cody: "Start over" (talks while he's typing, but it's unclear).
Boy left: "Now I feel like we're the ones teaching Cody here…"
Boy right: "…he didn't even do the height." (commenting on the code execution)
Boy left: "Ah, right… we have to ask him kind of concretely…"
Boy right (takes the Chromebook): "…can you make a simple code for a right-angled triangle…" (prompting Cody)
Boy left: "No, you have to explain that that, what do you call it, that top, and his top angle was pointing the wrong… way." (commenting on the code execution)

In Extract 4, the boys position Cody as a tutee rather than an all-knowing helper: they critically inspect his long codes ("why is he making it so complicated?"), predict errors before running them, and identify specific missing geometric properties such as the height and the top angle ("he didn't even do the height, and his top angle was pointing the wrong… way"). Instead of simply accepting or rejecting Cody's output, they use their own understanding of the intended right-angled triangle to correct Cody's reasoning ("now I feel like we're the ones teaching Cody here"), reformulating the prompt to ask for a "simple code" and to specify the problematic angle more precisely. This interaction thus narrows the gap between code and geometry because they use domain-specific properties such as base, height and top angle in increasingly explicit ways. Here, Cody's code again produces a geometrically incorrect shape relative to the task, which the students identify by comparing it with their own expectations about a right-angled triangle.

Across the four excerpts, our results show that students enact Cody in three different but intertwined ways. *First*, students sometimes use Cody as a tool—like a search engine or black-box code generator—to obtain names, facts, or ready-made code with minimal follow-up, so the intended geometric ideas are only weakly engaged. *Second*, in more extended, conversational uses, they iteratively prompt Cody, copy–paste–run its suggestions, and comment on the "weird" or "correct" shapes that appear; this can either narrow the gap between code and geometry or widen it. When the students focus mainly on getting something to work, are deeply engaged, but do not listen to the teacher, the gap is widened as we showed in Extract 3. However, when they are helped by the teacher, the gap can be narrowed. *Third*, in the most advanced cases, students narrowed the gap themselves by moving to a higher level of abstraction by comparing Cody's code with their own and with the task, treating Cody's output and explanations as objects of empirical inquiry. The students become more comfortable and begin to "help" Cody by supplying precise geometric information, extending working from human–computer interaction to code–>AI–>domain object modification, which we interpret as a light form of conversational EUD as we elaborate below.

5. Discussion and conclusions

In this paper, we ask: How is the school-adapted chatbot Cody enacted as an AI assistant in middle school mathematics programming activities? Based on our selected episodes, we identified three intertwined roles for Cody—tool, conversational partner, and tutee/object of inquiry—articulated both implicitly and explicitly in students' talk (e.g., "now I feel like we're the ones teaching Cody here", "we had to help Cody with the angles"), and we use these roles as an analytic lens rather than as a complete typology of all AI use in the classroom [28, 27]. In the discussion, we compare these roles across four cross-cutting themes derived from the literature: (1) how the roles relate to existing typologies of AI assistance, (2) how they shape the relationship between code and geometry, (3) how they extend an inquiry perspective on programming, and (4) how they open possibilities for end-user development in the tutee role.

Our three Cody roles compare with and extend existing typologies of AI assistance in education. Holmes and Tuomi's [24] categories of AI for content delivery, tutoring, and orchestration capture Cody's tool role, but not the way Cody becomes a problematic collaborator or object of inquiry when students treat it as conversational partner or tutee. Design-oriented work on GenAI support [13, 23] consider AI as a largely beneficial scaffold for cognitive and metacognitive processes, whereas our data show how such support can oscillate, sometimes requiring students and teacher to "scaffold the AI" instead. Finally, in contrast to studies that emphasise productivity, feedback, or access in AI-supported writing and coding [14], our analysis foregrounds students' situated enactments of an AI assistant in code–geometry work, including moments where Cody itself becomes the target of diagnosis and repair rather than a stable source of help.

Our analysis of students' dialogs around code, turtle figures, and Cody echoes computer-supported collaborative learning (CSCL) work on collaborative exploration of dynamic geometry, where fine-grained interaction analysis shows how geometric meanings are negotiated in and through tool-mediated conversation [17]. In our case, this conversation includes an AI agent that can either compress or expand the distance between code and domain concepts, depending on how it is used. This raises an instructional design question: when do additional mediating tools (turtle, code, AI) help learners experience direct interaction with geometric ideas, and when do stacked representations risk a permanently wide gap where tool handling displaces domain understanding [16, 8, 21]. A concerns is that programming tools can displace mathematical focus [22, 21], which identify an area for further work on programming in mathematics education.

Our findings extend Litherland and Kluge's [5] view of programming as empirical inquiry by showing how the "conversation" with code becomes triadic once an AI assistant is in the loop. In Schön's [4] terms, they engage in a reflective conversation with the "materials" of the situation, where those materials now include text responses and logged interactions as well as code and turtle figures. This aligns with Fischer and Lemke's [29] call to shift attention from human–computer

interaction to human problem-domain communication: when students talk about Cody making "weird shapes" and work to fix them, they are really negotiating what counts as an acceptable triangle or polygon in geometric terms, using buggy outputs as occasions to articulate and refine their understanding of the problem domain. Over time, such work can help students connect these activities to more advanced geometric concepts, formulas (e.g., the Pythagorean theorem), and trigonometric functions (e.g., the cosine rule).

A design tension appears when some of the more active students try "helping Cody" rather than focusing in on geometric content. We did not anticipate the tutee role for Cody: we expected students to use the chatbot primarily as an interactive tutor or teaching assistant [3]. Thus, while the tutee role was enabled by the instructional design (open-ended tasks and invitations to "talk with Cody"), the specific ways in which students positioned Cody as someone they were "teaching" emerged in their own interactional work (e.g., Extract 4). What the students did with Cody when it acted as a tutee is an in-situ kind of tailoring. Students do not modify the underlying model nor create a locally trained version, but they iteratively customize Cody's visual behaviour within a single chat thread by specifying domain-knowledge constraints (angles, side lengths, shape properties), and rejecting unhelpful suggestions at both semantic (explicit) and latent (implicit) levels in their utterances.

This collective adaptation work starting in a school municipality and including a teacher and some of the students echoes Fischer and colleagues meta-design view of end users shaping a socio-technical environment through ongoing adaptation on multiple levels and roles [25, 1, 30]. We recognise that students working with Cody in the classroom engage in "lightweight EUD:" their changes are local and ephemeral, and the threshold for participation is very low. Precisely this easily reached but narrow scope, however, points to a design space that current overviews of EUD for AI [2] largely leave open. In our case, EUD takes the form of a conversational, classroom-embedded "micro-tailoring," where learners gradually learn to teach and correct an AI assistant as part of their domain inquiry using natural language interaction, without ever using explicit developer tools.

This constitutes a light form of EUD at the *interaction level* because students cannot change Cody's underlying model, but they do shape its behaviour within a session by specifying domain constraints and rejecting unsatisfactory outputs, which in turn has consequences for their mathematical sense-making. We therefore distinguish between *model-level* control (which students lack) and interaction-level control over Cody's suggestions, which is where their agency is exercised in our classroom data.

The AI-as-tutee dynamic therefore reflects genuine, but bounded, user agency, students decide which outputs to accept, how to re-specify constraints, and when to reject Cody's suggestions, while ultimate control over the model remains external. For learning, this creates opportunities for students to articulate geometric constraints through experimentation and reflective conversation, activities aligned with constructionist pedagogy, but also the risk that effort is spent on "helping Cody" rather than consolidating mathematical ideas, a design tension we identified above.

5.1. Limitations of the study and directions for further work

Our qualitative design involved selective sampling of episodes rather than a comprehensive account of all classroom activity. Like Litherland and Kluge [5], who use carefully chosen excerpts to foreground ways of "conversing" with code rather than to represent a whole population of learners, we selected interaction sequences that made Cody's three enactments (tool, partner, tutee) empirically visible and analytically tractable. This means that our material highlights illustrative and sometimes exceptional moments of Cody use, while less articulate, shorter, or more routine interactions are under-represented. Moreover, our data come from a small number of lessons occurring over two days in one Norwegian middle school context in an all-boy group, and prior work suggests that gender can shape experiences and participation in programming activities, so our findings may not capture possible gender-related differences.

Furthermore, some of the conversational episodes with Cody are only partially recoverable due to noisy classrooms and students forgetting to turn on screen recording or invertedly delete their

data, which may bias the analysis toward episodes where interaction and sense-making were more visible in the available records. Our analysis would have benefited from more persistent digital traces of interaction (full Cody logs), like the rich records used in CSCL studies of collaborative dynamic geometry [17]. Moreover, we did not systematically distinguish between model hallucinations and student misinterpretations of Cody's output, which future work could address by combining classroom video with full Cody logs. Future work should explore how Cody-like assistants are enacted across different tasks, subjects, teachers, schools, municipalities, and countries to test whether our tentative hypothesis of three types of GenAI assistance holds.

Acknowledgements

The researchers acknowledge the time and resources that teachers and senior advisors (Arne Bergan, Marie Olafsen and Rune Glad, Asker Municipality) have dedicated to the research presented in this paper. The Learning in the Age of Algorithms (LAT project) is funded by the Norwegian Research Council, Grant number 341216.

Disclosure of Interests. The authors have no competing interests to disclose.

Declaration on Generative AI

During the preparation of this work, the authors used GPT UiO (a version of ChatGPT 4.o adapted at University of Oslo) for grammar, spelling check, and formatting. After using this GenAI tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. Fischer, D. Fogli, A. Piccinno, Revisiting and broadening the meta-design framework for end-user development, in: F. Paternò, V. Wulf (Eds.), New Perspectives in End-User Development, Springer, 2017, pp. 61–97.

[2] A. Esposito, et al., End-user development for artificial intelligence: A systematic literature review, in: L.D. Spano, et al. (Eds.), Proceedings of the 9th International Symposium on End-User Development (IS-EUD 2023), Springer, 2023.

[3] A.I. Mørch, S. Ludvigsen, Ø. Gilje, Teachers as end-user developers: Two case studies of adapting language models for education, in: Joint Proceedings of IS-EUD 2025: 10th International Symposium on End-User Development, CEUR-WS, Munich, Germany, 2025, pp. 16–18.

[4] D.A. Schön, Designing as reflective conversation with the materials of a design situation, Knowledge-Based Systems 5 (1992) 3–14.

[5] K. Litherland, A. Kluge, Learning to program as empirical inquiry: Using a conversation perspective to explore student programming processes, Computer Science Education 34 (2024) 495–519. doi:10.1080/08993408.2023.2290410.

[6] J. Lundin, R. Andersen, T. Leinonen, N.B. Dohn, H. Swensen, L. Mifsud, et al., Nordic approaches to computational thinking in teaching and learning, in: Proceedings of the International Conference on Networked Learning, vol. 12, 2024, pp. 384–386.

[7] A. Mørch, Y. Kafai, Computational thinking as a social movement, KI – Künstliche Intelligenz 36 (2022) 87–90.

[8] S. Papert, I. Harel, Constructionism, in: I. Harel, S. Papert (Eds.), Constructionism, Ablex Publishing, 1991, pp. 1–11.

[9] L.S. Vygotsky, Mind in Society: The Development of Higher Psychological Processes, Harvard University Press, Cambridge, MA, 1978.

[10] K. VanLehn, The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems, Educational Psychologist 46 (2011) 197–221.

[11] A.A. diSessa, Computational literacy and "the big picture" concerning computers in mathematics education, Mathematical Thinking and Learning 20 (2018) 3–31.

[12] B.J. Reiser, Why scaffolding should sometimes make tasks more difficult for learners, in: Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community, International Society of the Learning Sciences, 2002, pp. 255–264.
[13] E. Kasneci, et al., ChatGPT for good? On opportunities and challenges of large language models for education, Learning and Individual Differences 103 (2023) 102274.
[14] M. Warschauer, et al., The affordances and contradictions of AI-generated text for writers of English as a second or foreign language, Journal of Second Language Writing 62 (2023) 101071.
[15] M. Resnick, Lifelong Kindergarten: Cultivating Creativity Through Projects, Passion, Peers, and Play, MIT Press, Cambridge, MA, 2017.
[16] A.A. diSessa, H. Abelson, Turtle Geometry: The Computer as a Medium for Exploring Mathematics, MIT Press, Cambridge, MA, 1986.
[17] G. Stahl, Constructing Dynamic Triangles Together: Student Collaboration in a Virtual Math Team, Springer, New York, NY, 2015.
[18] D. Wood, J.S. Bruner, G. Ross, The role of tutoring in problem solving, Journal of Child Psychology and Psychiatry 17 (1976) 89–100.
[19] A.I. Mørch, S. Jondahl, J.A. Dolonen, Supporting conceptual awareness with pedagogical agents, Information Systems Frontiers 7 (2005) 39–53.
[20] D. Wood, H. Wood, Vygotsky, tutoring and learning, Oxford Review of Education 22 (1996) 5–16.
[21] M. Misfeldt, U.T. Jankvist, E. Geraniou, K. Bråting, Relations between mathematics and programming in school: Juxtaposing three different cases, in: Proceedings of the 10th ERME Topic Conference on Mathematics Education in the Digital Age (MEDA 2020), European Society for Research in Mathematics Education (ERME), 2020, pp. 252–262.
[22] M.K. Fojcik, Norwegian mathematics teachers' conceptions of programming in mathematics education, LUMAT: International Journal on Math, Science and Technology Education 13 (2025) 19. https://doi.org/10.31129/LUMAT.13.1.2936
[23] I. Molenaar, Towards hybrid human–AI learning technologies, European Journal of Education 57 (2022) 632–645.
[24] W. Holmes, I. Tuomi, State of the art and practice in AI in education, European Journal of Education 57 (2022) 542–570.
[25] G. Fischer, A conceptual framework for computer-supported collaborative learning at work, in: S. Goggins, I. Jahnke, V. Wulf (Eds.), Computer-Supported Collaborative Learning at the Workplace, Computer-Supported Collaborative Learning Series, vol. 14, Springer, 2013, pp. 23–42.
[26] N.K. Denzin, The Research Act: A Theoretical Introduction to Sociological Methods, 1st ed., Routledge, New York, NY, 2009.
[27] F. Erickson, Qualitative methods in research on teaching (Occasional Paper No. 81), Institute for Research on Teaching, Michigan State University, East Lansing, MI, 1985.
[28] V. Braun, V. Clarke, Using thematic analysis in psychology, Qualitative Research in Psychology 3 (2006) 77–101.
[29] G. Fischer, A.C. Lemke, Construction kits and design environments: Steps toward human problem-domain communication, Human–Computer Interaction 3 (1987) 179–222.
[30] A.I. Mørch, J.E.B. Nævdal, Helping users customize their pedagogical agents, in: V. Palade, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems: KES 2004, Lecture Notes in Artificial Intelligence, vol. 3213, Springer, Berlin, 2004, pp. 131–139.