Pedagogy for critical engagement with Generative AI

James Prather
Abilene Christian University
Abilene, Texas, USA
james.prather@acu.edu

Brent N. Reeves
Abilene Christian University
Abilene, Texas, USA
brent.reeves@acu.edu

ABSTRACT

The advent of generative AI (GenAI) systems is radically shifting Computer Science education. Early models in 2021 could already solve most introductory problems and modern models are expanding that to more complex and advanced areas of the discipline. Educators are worried about student over-reliance on GenAI tools and their potential for undercutting critical thinking. Although current research shows these fears are not unfounded, it also shows a way forward. A new pedagogical approach is needed for teaching Computer Science with AI that protects human cognition, augments current processes, and offloads the unnecessary.

KEYWORDS

metacognition, computer science education, novice programmers, GenAI, code generation tools

ACM Reference Format:
James Prather and Brent N. Reeves. 2025. Pedagogy for critical engagement with Generative AI . In ,. ACM, New York, NY, USA, 4 pages. https://doi.org/11.1111/1111111.1111111

1 INTRODUCTION

1.1 The Trade-offs of GenAI for Novice Programmers

Researchers have been exploring both the opportunities and challenges inherent in GenAI tools, such as Gemini, ChatGPT, and GitHub Copilot since 2021.

GenAI tools can benefit instructors by offloading some tedious work. Early research in computing education showed that GenAI could be used to generate customized programming exercises. These models performed so well that they could be provided to students [44]. A followup study two years later showed that allowing students to customize these assignments created more engagement with programming activities [27]. Other work showed that GenAI could create multiple choice questions for exams in computing courses [1, 10, 10, 50].

Students can also benefit from using them, helping them better understand difficult concepts such as recursion [4], program code [19, 23, 31, 44], and error messages [24, 43, 48, 52]. Students rate GenAI explanations as helpful [31] and large scale implementations show them to be effective at reducing rates of repeated errors [48, 52]. Very recent work has shown that it may not quite be the silver bullet yet, and that human explanations can be better, but only by experts [43]. Students have also reported high success rates at using GenAI tools for help-seeking and feedback on programs [15, 20, 21]. Finally, GenAI can be used to let students prompt for code generation in their native languages, breaking down traditional barriers around the English language and broadening participation globally in radically new ways [40].

However, there are many difficulties that researchers have also discussed, often in the same work that shows these opportunities. The first paper on the subject, with data collected in July of 2021, showed that GenAI could already correctly generate code to an introductory programming problem that typically stumps novices [12] and could correctly answer most exam questions. Subsequent work found that later models could handle more advanced questions from CS2 [13] and algorithms [14]. A replication of that first study with more advanced models showed drastically improved performance [35]. Further research has shown GenAI models can solve different types of problems, such as Parsons Problems [42]. Multimodal models, which contain visual and spacial reasoning skills, can solve Parsons problems from images [16] and computer graphics questions [11].

These capabilities have led to educator fears that students will become over-reliant on them [2, 22]. Recent work has shown that novices with low self-efficacy will use GenAI earlier in the problem solving process and more often throughout the semester [32]. GenAI coding tools can also be difficult for novices who have not developed metacognitive programming skills [41]. Underprepared, unconfident, and underperforming students seem to benefit the least from these tools [17] resulting in less critical thinking and lower grades [18].

1.2 New GenAI Pedagogies

Even as traditional programming skills are reevaluated in the context of GenAI capabilities, new skills are emerging, such as prompt engineering. This skill can be applied both to code generation and code comprehension.

Very early on after the advent of GenAI, researchers noticed that writing prompts well resulted in better outcomes for students, both in their engagement and in the accuracy of the generated code [6]. Many have since argued that learning how to craft an effective prompt is an emerging skill for programmers and should be explicitly taught to novices alongside traditional ones [8]. In line with this, researchers introduced "Prompt Problems" as a novel pedagogical intervention for novice programmers that could teach both at the same time [7]. Prompt Problems are presented visually for a student to decompose into a prompt that could generate code to solve the problem depicted in the image. The prompt is used to generate code that is then tested against a suite of test cases. If it

CHI, CHI Conference on Computer-Human Interaction

ACM ISBN 111-1-111-1111-1/11/11
https://doi.org/11.1111/1111111.1111111

does not pass, the student reads the failing test case and the generated code, and then determines how to edit their prompt. Students persist in this cycle until the problem is complete. Students reported that the activity was enjoyable and caused critical reflection on the problem at a high level [7, 36].

Another GenAI-powered approach is the use of Explain in Plain English (EiPL) problems at scale. These questions require students to read a piece of code and then explain in clear and simple terms what it does. Higher grades are awarded to students who talk about the code at a high level, while lower marks are awarded to students who discuss it at a line-by-line level. Much work in computing education research has been conducted on EiPL showing that student programming ability is connected to their ability to clearly explain code [5, 26, 30, 51, 53]. However, the primary difficulty in using EiPL questions has been in grading them at scale. Using GenAI to grade EiPL problems has recently been shown to be effective [9, 46]. This is done by using the student code explanation as a prompt to a GenAI model to generate code. That code is then run against test cases, similar to a Prompt Problem, and feedback provided to students based on test case coverage. Connecting prompt writing and code comprehension tasks has been shown to benefit students and help them to connect the prompts they write to the code that is generated [45].

Recent work implemented both EiPL questions and Prompt Problems into an introductory programming course and found that students who struggle with traditional coding tasks performed better when completing the questions involving prompting via natural language. This suggests that such tasks engage a wider range of cognitive skills and offer a more accessible starting point for novice programmers.

2 DISCUSSION: CONSIDERATIONS FOR DESIGNING A NEW PEDAGOGY

From the review above, it is evident that a new pedagogical approach is needed for teaching Computer Science with GenAI. This new approach should protect human cognition by teaching students to critically engage with AI-generated code, augment current processes by helping students solve problems more efficiently, and offload the unnecessary through a chance to refocus our curricula.

2.1 Protecting Human Cognition

A GenAI pedagogy must help students learn how to critically evaluate AI-generated content without undercutting their critical thinking skills [18, 41]. Even though code generation tools are getting better, there are still performance trade-offs, biases in generated content from the model training data, security concerns, and more, in generated code. Novices put too much faith in GenAI coding tools and often passively accept generated code, assuming it is what they need [41]. Instead, students must be trained to recognize and mitigate these harms through critical evaluation of generated code. Approaches like Prompt Problems [7] and EiPL questions [9] are a first step toward building these awarenesses in students.

2.2 Augmenting Current Processes

Novice programmers have historically struggled with metacognition [28, 29, 34, 38] and recent research shows that GenAI can make it worse [41]. This is partly because GenAI interfaces are often not designed to support metacognition [47]. Novices especially struggle with information overload from code they didn't write and can't understand. This is intimidating and leaves novices unsure when to accept generated code and when to persist on their own path. The source code that is suggested in the editing window seems to confuse them more than help them. Yet research shows that successful novices do learn when to decline code suggestions. They do not become overwhelmed by the code suggestions. They are able to "shrug off" suggestions that don't seem to be leading them towards their goal [39, 41]. This "negative expertise", a concept first proposed by Marvin Minsky, allows programmers to understand what to do as much as what *not* to do [33]. While recent research has noted that some novices are better at this than others [41], more work is needed on how to directly scaffold this metacognitive skill in novices. By carefully scaffolding the coding experience for novices, Gen AI can help to expose to them their own hidden metacognitive processes.

2.3 Offloading the Unnecessary

The fact that a simple prompt can generate quite powerful or complicated code calls us to reflect on pedagogy with a critical eye towards eliminating the things that might have been important at the time, but no longer add value to a student. Like unhelpful, esoteric, and difficult compiler error messages, which were taught because they stood in the way of a successful program [3]. GenAI appears to have solved one of the longest standing problems in computing education – and perhaps computing broadly construed. Because research shows GenAI can not only explain, but also fix most syntax and even runtime errors [25, 49, 52], does this mean we should stop teaching them?

Similarly, it's possible that current students may not need to have as deep a grasp of programming language syntax as their previous peers because the initial generation of code is not done by the human, but rather with GenAI [37]. Our curricula were tuned to the errors inevitably encountered in student-written code. But many of those errors will never be seen by today's students. Even some semantic errors could be a thing of the past, such as off-by-one errors, because GenAI may not ever write a loop in the first place that exceeds the bounds of a list.

3 OPEN PROBLEMS AND FUTURE WORK

Many challenges remain in this new and rapidly shifting landscape. While it is unclear how all of this will shake out over the next decade, we believe there are some clear problems that must be solved and a clear direction for future work.

3.1 Open Problems

Broadly speaking, we see three open problems that must be solved.

3.1.1 Outsourcing the "wrong" cognitive tasks: The ability to offload some tasks to AI fundamentally alters how we teach code and prepare students for their work. However, it may be that certain tasks that we have always had students perform in fact accidentally serve deeper cognitive purposes. For instance, debugging may have been a frustrating and seemingly cognitively useless task that students are freed from with generative AI, but it may also have fostered

positive cognitive habits such as perseverance, attention to detail, and problem solving. Choosing which tasks to discard will be of important moving forward, and will take careful investigation to determine if how the loss traditional programming tasks is associated with valuable critical thinking and abstraction skills.

3.1.2 The art of design: Students can easily create large code bases and move quickly through modification and toward completion. This advance surely opens opportunities to do more, faster. However, it's possible that moving the level of design to natural language removes some of the joy of tackling complexity. One of the primary joys of computing disciplines is in creative problem solving. Edsger W. Dijkstra once wrote that "Much of the excitement we get out of our work is that we don't really know what we are doing." Therefore, we must determine what levels of design interaction are ideal for learning or productivity, and enhance those while discarding others.

3.1.3 Less is more (maybe): As students become more familiar with AI coding tools, they will be able to modify and refactor existing code bases with increasing ease. However, it's possible that the persistent use of LLMs shield the user from the experience of having to remove parts of a design that were hard-fought, thus robbing the learner from lessons learned in abstractions and refactoring. Solving the initial problem is only the beginning in real code bases and there are many important decisions to be made and problems to be solved. A narrow focus on initial problem solving without a wider view of the life of code could inadvertently introduce a decline in skills related to code management, refactoring, and extension.

3.2 Future Work

We see a shift in programming education moving forward. A full integration of GenAI would mean teaching it from the first day of the first course. In a similar vein to the old "objects first" debate, we propose a "prompts first" approach to teaching and learning programming moving forward. Using a "prompts first" approach, students begin on day one by reading small snippets of code for comprehension but also writing prompts to generate code, such as the classic "Hello, World!" program. Students build competency in abstraction and problem solving while still building a vocabulary of syntax, ideas of control flow, and skill in prompt writing all while doing so in their native language. As problems increase in complexity, so also do the challenge of decomposing, specifying, and verifying.

Implementing a "prompts first" approach is daunting and will require research. What tasks are best to help students learn both prompting *and* traditional programming concepts? How can we verify that students legitimately understand prompting and programming without developing over-reliance? What does a "prompts first" CS1 course look like? And how does this impact the subsequent curriculum? What does a software engineering course in a "prompts first" curriculum look like? And how can all of this protect and augment human cognition, rather than cede it to the machines? A research agenda centering around these questions is beginning to emerge that will shape the next decade of human-computer interaction research in the domain of computer science education.

REFERENCES

[1] Arav Agarwal, Karthik Mittal, Aidan Doyle, Pragnya Sridhar, Zipiao Wan, Jacob Arthur Doughty, Jaromir Savelka, and Majd Sakr. 2024. Understanding the Role of Temperature in Diverse Question Generation by GPT-4. In *Proc. of the 55th ACM Technical Symposium on Computer Science Education V. 2*. 1550–1551.

[2] Brett A Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard–Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1.*

[3] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. In *Proc. of the Working Group Reports on Innovation and Technology in Computer Science Education*. ACM, NY, NY, USA, 177–210.

[4] Seth Bernstein, Paul Denny, Juho Leinonen, Lauren Kan, Arto Hellas, Matt Littlefield, Sami Sarsa, and Stephen Macneil. 2024. "Like a Nesting Doll": Analyzing Recursion Analogies Generated by CS Students Using Large Language Models. In *Proc. of the 2024 on Innovation and Technology in Computer Science Education V. 1*. ACM, NY, NY, USA, 122–128.

[5] Malcolm Corney, Sue Fitzgerald, Brian Hanks, Raymond Lister, Renee McCauley, and Laurie Murphy. 2014. 'Explain in Plain English' Questions Revisited: Data Structures Problems. In *Proc. of the 45th ACM Technical Symposium on Computer Science Education*. ACM, NY, NY, USA, 591–596.

[6] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, NY, USA, 1136–1142.

[7] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proc. of the 55th ACM Technical Symposium on Computer Science Education V. 1*. ACM, NY, NY, USA, 296–302.

[8] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. 2024. Computing Education in the Era of Generative AI. *Commun. ACM* 67, 2 (jan 2024), 56–67.

[9] Paul Denny, David H Smith IV, Max Fowler, James Prather, Brett A Becker, and Juho Leinonen. 2024. Explaining code with a purpose: An integrated approach for developing code comprehension and prompting skills. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. 283–289.

[10] Jacob Doughty, Zipiao Wan, Anishka Bompelli, Jubahed Qayum, Taozhi Wang, Juran Zhang, Yujia Zheng, Aidan Doyle, Pragnya Sridhar, Arav Agarwal, et al. 2024. A comparative study of AI-generated (GPT-4) and human-crafted MCQs in programming education. In *Proc. of the 26th Australasian Computing Education Conf.* 114–123.

[11] Tony Haoran Feng, Paul Denny, Burkhard C Wünsche, Andrew Luxton-Reilly, and Jacqueline Whalley. 2024. An Eye for an AI: Evaluating GPT-4o's Visual Perception Skills and Geometric Reasoning Skills Using Computer Graphics Questions. In *SIGGRAPH Asia 2024 Educator's Forum* (Tokyo, Japan) *(SA Educator's Forum '24)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3680533.3697064

[12] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The Robots are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proceedings of the 24th Australasian Computing Education Conference*. 10–19.

[13] James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, James Prather, and Brett A Becker. 2023. My AI Wants to Know If This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In *Proceedings of the 25th Australasian Computing Education Conference*. 97–104.

[14] Sebastian Gutierrez, Irene Hou, Jihye Lee, Kenneth Angelikas, Owen Man, Sophia Mettille, James Prather, Paul Denny, and Stephen MacNeil. 2024. Seeing the Forest and the Trees: Solving Visual Graph and Tree Based Data Structure Problems using Large Multimodal Models. *arXiv preprint arXiv:2412.11088* (2024).

[15] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proc. of the 2023 ACM Conf. on Int. Computing Education Research - Volume 1*. ACM, NY, NY, USA, 93–105.

[16] Irene Hou, Owen Man, Sophia Mettille, Sebastian Gutierrez, Kenneth Angelikas, and Stephen MacNeil. 2024. More Robots are Coming: Large Multimodal Models (ChatGPT) can Solve Visually Diverse Images of Parsons Problems. In *Proc. of the 26th Australasian Computing Education Conf.* ACM, NY, NY, USA, 29–38.

[17] Irene Hou, Sophia Mettille, Owen Man, Zhuo Li, Cynthia Zastudil, and Stephen MacNeil. 2024. The Effects of Generative AI on Computing Students' Help-Seeking Preferences. In *Proc. of the 26th Australasian Computing Education Conf.* 39–48.

[18] Gregor Jošt, Viktor Taneski, and Sašo Karakatič. 2024. The Impact of Large Language Models on Programming Education and Student Learning Outcomes. *Applied Sciences* 14, 10 (2024), 4115.

[19] Breanna Jury, Angela Lorusso, Juho Leinonen, Paul Denny, and Andrew Luxton-Reilly. 2024. Evaluating LLM-generated Worked Examples in an Introductory Programming Course. In *Proceedings of the 26th Australasian Computing Education Conference* (Sydney, NSW, Australia) *(ACE '24)*. Association for Computing Machinery, New York, NY, USA, 77–86. https://doi.org/10.1145/3636243.3636252

[20] Natalie Kiesler, Dominic Lohr, and Hieke Keuning. 2023. Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. *arXiv preprint arXiv:2309.00029* (2023).

[21] Charles Koutcheme, Nicola Dainese, Sami Sarsa, Arto Hellas, Juho Leinonen, and Paul Denny. 2024. Open Source Language Models Can Provide Feedback: Evaluating LLMs' Ability to Help Students Using GPT-4-As-A-Judge. In *Proc. of the 2024 on Innovation and Technology in Computer Science Education V. 1.* 52–58.

[22] Sam Lau and Philip J Guo. 2023. From" Ban It Till We Understand It" to" Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. *The 19th ACM Conf. on Int. Computing Education Research (ICER)* (2023).

[23] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *Proc. of the 2023 Conf. on Innovation and Technology in Computer Science Education V. 1.* ACM, NY, NY, USA, 124–130.

[24] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1.* ACM.

[25] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 1.* ACM, NY, NY, USA, 563–569.

[26] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further Evidence of a Relationship between Explaining, Tracing and Writing Skills in Introductory Programming. In *Proc. of the 14th Annual ACM SIGCSE Conf. on Innovation and Technology in Computer Science Education.* ACM, NY, NY, USA, 161–165.

[27] Evanfiya Logacheva, Arto Hellas, James Prather, Sami Sarsa, and Juho Leinonen. 2024. Evaluating contextually personalized programming exercises created with generative AI. In *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1.* 95–113.

[28] Dastyni Loksa, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. 2016. Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. In *Proc. of the 2016 CHI Conf. on Human Factors in Computing Systems.* ACM, NY, NY, USA, 1449–1461.

[29] Dastyni Loksa, Lauren Margulieux, Brett A Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. 2022. Metacognition and self-regulation in programming education: Theories and exemplars of use. *ACM Transactions on Computing Education (TOCE)* 22, 4 (2022), 1–31.

[30] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships between Reading, Tracing and Writing Skills in Introductory Programming. In *Proc. of the Fourth Int. Workshop on Computing Education Research.* ACM, NY, NY, USA, 101–112.

[31] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 1.* ACM, NY, NY, USA, 931–937.

[32] Lauren E Margulieux, James Prather, Brent N Reeves, Brett A Becker, Gozde Cetin Uzun, Dastyni Loksa, Juho Leinonen, and Paul Denny. 2024. Self-Regulation, Self-Efficacy, and Fear of Failure Interactions with How Novices Use LLMs to Solve Programming Problems. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1.* 276–282.

[33] Marvin Lee Minsky. 1994. Negative Expertise. *International Journal of Expert Systems Research and Applications* 7 (1994), 13–18. Issue 1.

[34] James Prather, Brett A Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. 2020. What do we think we think we are doing? Metacognition and self-regulation in programming. In *Proceedings of the 2020 ACM conference on international computing education research.* 2–13.

[35] James Prather, Paul Denny, Juho Leinonen, Brett A Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, et al. 2023. The robots are here: Navigating the generative ai revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education.* 108–159.

[36] James Prather, Paul Denny, Juho Leinonen, David H Smith IV, Brent N Reeves, Stephen MacNeil, Brett A Becker, Andrew Luxton-Reilly, Thezyrie Amarouche, and Bailey Kimmel. 2024. Interactions with prompt problems: A new way to teach programming with large language models. *arXiv preprint arXiv:2401.10759* (2024).

[37] James Prather, Juho Leinonen, Natalie Kiesler, Jamie Gorson Benario, Sam Lau, Stephen MacNeil, Narges Norouzi, Simone Opel, Vee Pettit, Leo Porter, et al. 2025. Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools. *2024 Working Group Reports on Innovation and Technology in Computer Science Education* (2025), 300–338.

[38] James Prather, Raymond Pettit, Kayla McMurry, Alani Peters, John Homer, and Maxine Cohen. 2018. Metacognitive difficulties faced by novice programmers in automated assessment tools. In *Proceedings of the 2018 ACM Conference on International Computing Education Research.* 41–50.

[39] James Prather, Brent N. Reeves, Paul Denny, Brett A. Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Trans. Comput.-Hum. Interact.* 31, 1, Article 4 (nov 2023), 31 pages.

[40] James Prather, Brent N Reeves, Paul Denny, Juho Leinonen, Stephen MacNeil, Andrew Luxton-Reilly, João Orvalho, Amin Alipour, Ali Alfageeh, Thezyrie Amarouche, et al. 2024. Breaking the Programming Language Barrier: Multilingual Prompting to Empower Non-Native English Learners. *arXiv preprint arXiv:2412.12800* (2024).

[41] James Prather, Brent N Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S Randrianasolo, Brett A Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. 2024. The widening gap: The benefits and harms of generative ai for novice programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1.* 469–486.

[42] Brent Reeves, Sami Sarsa, James Prather, Paul Denny, Brett A Becker, Arto Hellas, Bailey Kimmel, Garrett Powell, and Juho Leinonen. 2023. Evaluating the performance of code generation models for solving Parsons problems with small prompt variations. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1.* 299–305.

[43] Eddie Antonio Santos and Brett A Becker. 2024. Not the Silver Bullet: LLM-enhanced Programming Error Messages are Ineffective in Practice. *arXiv preprint arXiv:2409.18661* (2024).

[44] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proc. of the 2022 ACM Conf. on Int. Computing Education Research - Volume 1.* ACM, NY, NY, USA, 27–43.

[45] David H. Smith, Paul Denny, and Max Fowler. 2024. Prompting for Comprehension: Exploring the Intersection of Explain in Plain English Questions and Prompt Writing. In *Proc. of the Eleventh ACM Conf. on Learning @ Scale.* ACM, NY, NY, USA, 39–50.

[46] David H. Smith and Craig Zilles. 2024. Code Generation Based Grading: Evaluating an Auto-grading Mechanism for "Explain-in-Plain-English" Questions. In *Proc. of the 2024 on Innovation and Technology in Computer Science Education V. 1.* ACM, NY, NY, USA, 171–177.

[47] Lev Tankelevitch, Viktor Kewenig, Auste Simkute, Ava Elizabeth Scott, Advait Sarkar, Abigail Sellen, and Sean Rintel. 2024. The Metacognitive Demands and Opportunities of Generative AI. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 680, 24 pages. https://doi.org/10.1145/3613904.3642902

[48] Andrew Taylor, Alexandra Vassar, Jake Renzella, and Hammond Pearce. 2024. dcc --help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models. In *Proc. of the 55th ACM Technical Symposium on Computer Science Education V. 1.* 1314–1320.

[49] Andrew Taylor, Alexandra Vassar, Jake Renzella, and Hammond Pearce. 2024. dcc –help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models. In *Proc. of the 55th ACM Technical Symposium on Computer Science Education V. 1.* ACM, NY, NY, USA, 1314–1320.

[50] Andrew Tran, Kenneth Angelikas, Egi Rama, Chiku Okechukwu, David H. Smith, and Stephen MacNeil. 2023. Generating multiple choice questions for computing courses using large language models. In *2023 IEEE Frontiers in Education Conf.* IEEE, 1–8.

[51] Anne Venables, Grace Tan, and Raymond Lister. 2009. A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer. In *Proc. of the Fifth Int. Workshop on Computing Education Research Workshop.* ACM, NY, NY, USA, 117–128.

[52] Sierra Wang, John Mitchell, and Chris Piech. 2024. A large scale RCT on effective error messages in CS1. In *Proc. of the 55th ACM Technical Symposium on Computer Science Education V. 1.* 1395–1401.

[53] Jacqueline L. Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, P. K. Ajith Kumar, and Christine Prasad. 2006. An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. In *Proc. of the 8th Australasian Conf. on Computing Education - Volume 52.* Australian Computer Society, Inc., AUS, 243–252.