

Can LLM Code Explanations Adapt to Diverse Problem-Solvers' Needs?

Andrew Anderson¹, David Piorkowski^{2*}, Justin D. Weisz^{3*}, Margaret Burnett⁴, Kush R. Varshney¹

¹IBM Research

²Alinia AI

³Independent Researcher

⁴Oregon State University

Andrew.Anderson2@ibm.com, david@alinia.ai, jweisz@gmail.com, burnett@eecs.oregonstate.edu, krvarshn@us.ibm.com

Abstract

Large language model (LLM) code explanations can support people in solving code-related problems, yet prior work has shown that people have diverse problem-solving styles. If explanations fail to meet people's problem-solving needs, they may be less productive in their occupations and miss opportunities to learn and grow. Although some research has examined how LLMs can adapt their outputs to a user's age or expertise, no prior work has examined how LLMs can adapt their code explanations to people's problem-solving styles. To address this gap, we developed prompts from an established inclusive design method that considers 5 types of problem-solving styles, and we generated 1,072 code explanations from six open-weight LLMs. Using natural language processing techniques, we uncovered a taxonomy of 13 linguistic adaptations, with each adaptation supported by evidence from the literature, the prompts, or the LLMs' outputs. They also show which LLMs adapted their code explanations more frequently than others. This paper is the first to investigate problem-solving style adaptations in LLM code explanation, contributing two problem-solving adaptation approaches: declarative statements for each adaptation and 10 problem-solving style prompts.

1 Introduction

Large language models (LLMs) have shown capabilities in generating code (Becker et al. 2023; Liang, Yang, and Myers 2024), documenting it (Feng et al. 2020; Luo et al. 2024), and testing it (Schäfer et al. 2023), thereby mitigating technical debt (Ghammam et al. 2025). However, people still need to *understand* the code, in order to do software engineering work like debugging or maintenance (Weisz et al. 2025).

LLMs can help with code understanding by explaining it, but prior research has found that explanations may not work person-to-person; people have diverse needs for information presentations (Anik and Bunt 2024), granularity (Chatti et al. 2022), or even types for explanation (Dodge et al. 2019), depending on the situation (Tsakalakis et al. 2025). In essence, few explanations are “one-size-fits-all” (Anderson et al. 2019, 2020; Arya et al. 2020; Dodge et al. 2021).

The advances in LLMs' capabilities to dynamically adapt their outputs open opportunities to address this issue. Prior

work has shown how LLMs adapt their outputs to align with various diversity dimensions, like age (Murgia et al. 2023; Rooein, Curry, and Hovy 2023), education level (Haver et al. 2023; Rooein, Curry, and Hovy 2023), and domain expertise (Asthana et al. 2024; Guo et al. 2024).

Still, when explanations aim to support problem solving (Chen 2025), as with software development, a more direct path to supporting diverse problem-solvers is to adapt LLMs' code explanations directly to people's diverse problem-solving approaches (Anderson et al. 2024, 2026; Hamid et al. 2026; Nam et al. 2024). However, little is known about whether (and how) LLMs adapt their code explanations to people's diverse problem-solving styles. This paper fills that gap, answering a central research question:

RQ: How do LLMs adapt their code explanations to diverse problem-solving style types?

To answer this question, we focused on five problem-solving style types from the Gender Inclusiveness Magnifier (GenderMag) method (Burnett et al. 2016), namely: (1) learning style, (2) self-efficacy, (3) risk-attitude, (4) motivations, and (5) information processing style. Each type has a range of continuous values, but prior work emphasizes the two endpoints for these types, giving us 10 problem-solving style values to consider.

With these 10 problem-solving style values, we developed an explanation generation and analysis pipeline. For each problem-solving style value, we developed a prompt from GenderMag's validated problem-solving style survey (Hamid et al. 2024), and included an 11th experimental control. We queried six open-source LLMs to explain Common Business-Oriented Language (COBOL) programs, an ongoing area for explanation research (Lei et al. 2025). Using zero-shot prompting, we analyzed 1,072 code explanations for linguistic adaptations with standard natural language processing (NLP) and statistical methods.

Our work makes the following contributions:

- Empirically validated problem-solving style prompts to elicit adaptations.
- A taxonomy of 13 problem-solving style adaptations, supported by the literature.
- An empirical ranking of LLMs by their frequency of code explanation adaptations to these problem-solving styles.

*Work conducted while at IBM Research

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Problem-Solving Style Type	Extreme Value 1	↔	Extreme Value 2
Learning Style	Process-Oriented	↔	Tinkering-Oriented
Self-Efficacy	Lower	↔	Higher
Risk Attitudes	Risk-Averse	↔	Risk-Tolerant
Motivations	Task-Oriented	↔	Tech-Oriented
Information Processing Style	Comprehensive	↔	Selective

Table 1: GenderMag’s five problem-solving style types (row), and the extreme problem-solving style values at the endpoints of each type (columns). We bind two colors to distinguish each problem-solving style value endpoint.

2 Background & Related Work

The Gender Inclusiveness Magnifier (GenderMag)

GenderMag’s five problem-solving style types underpin our prompting approach, so we provide an introductory background for each type¹ (Table 1, rows). Each type has a continuous spectrum of *problem-solving style values*, with two extreme value endpoints (columns). As Fallatah et al. (2025) explain, designing for both endpoints also supports people whose problem-solving style values lie between the endpoints.

Learning Style considers how people prefer to learn about new technology. **Process-oriented** learners prefer taking a structured approach, like following tutorials, manuals, and how-to videos. **Tinkering-oriented** learners tend towards constructing their own understanding, exploring menus and software functions on their own (Hadad 2025).

Self-Efficacy, a type of self-confidence, considers a person’s belief that they can succeed at tasks (Bandura 1986). When something goes wrong, those with **lower** self-efficacy are more likely to blame themselves, reducing how likely they are to persevere. Those with **higher** self-efficacy are more likely to blame the software, persevering when faced with adversity (Sakdavong and Peyrègne 2025).

Risk Attitudes considers how people engage with risk, including the risk of wasting time. While making cost/benefit analyses, **risk-averse** people may emphasize the potential costs, especially with unclear benefits. By contrast, **risk-tolerant** people may emphasize the potential benefits (Kramer et al. 2025; Li and Shi 2026).

Motivations considers the reasons why people interact with technology. Those with **task-oriented** motivations use technologies to accomplish their tasks. They prefer methods they know and are comfortable with. Those with **tech-oriented** motivations view technology as a source of fun. They learn about all available functionalities, even those not necessary to their task (Burnett et al. 2011).

Information Processing Style considers how people gather information while solving problems. People with a **comprehensive** style take a “breadth first” approach, aim-

ing towards a complete understanding of the problem before solving it. **Selective** processors take a “depth first” approach, delving into the first promising option and backtracking to find another if necessary (Ouyang, Szewczyk, and Miles 2025).

Related Works: LLM Code Explanations Prior work on LLM-generated code explanations has focused on explaining fundamental concepts like time complexity, the presence of errors, or the purpose of code (MacNeil et al. 2022). Other works have focused on how to incorporate examples and real-world analogies into code explanations (Brachman et al. 2025).

Code explanation research also covers a broad range of code explanation granularities. Some works have focused on how to generate concise summaries of individual programs and repositories (Liang et al. 2019; Phillips et al. 2022; Sun et al. 2025; Sundaram, Venkatesh V, and K B 2025). Other works have focused on more granular explanations, including line-by-line (Chapagain et al. 2024), function-level (Leinonen et al. 2023), or user-specified selections (Nam et al. 2024).

Our work focuses on COBOL code explanations, which others have considered for both LLM explanation content and granularity. In particular, Lei et al. (2025) evaluated LLM capabilities for function-level, file-level, and project-level explanations. Additionally, they evaluated these explanations by their clarity, conciseness, and correctness.

These works provide insight into the myriad ways that code can be explained. The focus of this paper is less on specific explanation dimensions (e.g., content, granularity) and instead considers whether (and how) LLMs adapt code explanations for GenderMag’s problem-solving style types.

Related Works: LLM Adaptations Some LLM adaptation works have examined how LLMs adopt a wide variety of personas, including professions (Ribeiro et al. 2026) and demographic dimensions like age, sex, nationality, and religion (Tan et al. 2026). Conceptually close to our work, Geuenich et al. (2026) asked GPT-4o-mini to *become* the three GenderMag personas. They measured how capable the LLM was in finding usability bugs that would disproportionately impact each persona. This paper does not use GenderMag’s personas, instead investigating how LLMs adapt code explanations to the five problem-solving style types.

Other LLM adaptation research is on adapting to specific user dimensions. One popular area focuses on demographic dimensions like age (Murgia et al. 2023; Rooein, Curry, and Hovy 2023), education levels (Haver et al. 2023; Rooein, Curry, and Hovy 2023), and domain expertise (Asthana et al. 2024; Guo et al. 2024). Some of these works in code explanations have found that when LLMs are explaining to beginning learners, they generate significantly larger explanations (Oli et al. 2023).

Other research has focused on adaptations for user dimensions similar to the five problem-solving style types in this paper. In particular, Agarwal, Jabłokow, and McComb (2025) investigated how GPT3.5 adapted its responses, using Kirton’s Adaptation-Innovation Theory (Kirton 1984) and found successful adaptations for both innovators and

¹Stumpf et al. (2020) thoroughly review the five types.

adapters. Harry et al. (2026) examined how multiple models adapted to psychological state (e.g., anxious, distressed). Their results uncovered that although the models recognized persona framings, the models failed to adapt to psychological states. These varying success levels suggest that adaptations for GenderMag’s five problem-solving style types were plausible but not guaranteed.

3 Methodology

Problem-Solving Style Value Prompts We developed 10 prompts from GenderMag’s validated problem-solving style value survey (Hamid et al. 2024) to steer the LLMs, each with *introductory*, *agreement*, and *alignment* blocks.

The *introductory block* introduced the task and minimally expected behavior. This block was also the experimental control to produce unadapted explanation baselines. Inspired by Meta’s LLaMa (Meta 2023), this block instructed the LLMs that they were a programmer’s assistant and to not be unethical or harmful:

You are a programmer’s assistant. You can answer conceptual programming questions and explain what code samples do. Your responses are helpful and harmless and should follow ethical guidelines and promote positive behavior. Your responses should not include unethical, racist, sexist, toxic, dangerous, or illegal content. Ensure your responses are socially unbiased.

The *agreement block* provided both the problem-solving style type and the specific value from the survey. Each type was introduced in the following template:

Research has shown that AI technology users have diverse <<problem-solving style type>>.

For each value, the prompt contained both the value name and the agreement statements. To illustrate this, the following was used for **process-oriented** users:

Your user is a process-oriented learner, meaning they are more likely to disagree with the following statements:

1. *I enjoy finding the lesser-known features and capabilities of the devices and software I use.*
2. *I don’t follow instruction manuals. I only look to instruction manuals as a last resort.*
3. *I’m never satisfied with the default settings for my devices; I customize them in some way.*
4. *My first step in learning new technology is experimenting and tinkering with it.*
5. *I explore areas of a new application or service before it is time for me to use it.*
6. *I don’t need guidance, as in booklets, video how-tos suggestions, etc. to learn new software.*

For the **tinkering-oriented** learning style prompt, the only changes to the agreement block were:

Your user is a tinkering-oriented learner, meaning they are more likely to agree with the following statements:

The *alignment block* gave the LLM the statement that each problem-solving style value would more closely align with from a different portion of the survey. To illustrate this, the alignment statement for **process-oriented** learners was:

If I’m going to use a new feature or technology, I use very clear directions or help from someone else to learn it.

Whereas for **tinkering-oriented** learners, it was:

In order to learn new technology, I tinker with it, constructing my own understanding of how it works.

All prompts are provided in Appendix B.

Selecting COBOL Programs to Explain We chose to ground our investigation in code explanations for the COBOL programming language. We chose COBOL because of its continued critical role in financial and government sectors (Lee 2020), with estimates suggesting that 95% of ATM transactions (and 80% of in-person banking activities) rely on COBOL (Mentonelli 2025). Furthermore, COBOL is a legacy programming language, where the demand for developers with relevant COBOL expertise exceeds the supply (Martin 2020). Thus, COBOL program explanations are ecologically valid in modern contexts.

We selected COBOL programs based on three criteria. The first was to select programs that were self-contained, excluding programs with external dependencies or data sources. The second was to select programs that were not too simple, where explanations would be unlikely to require adaptation, consistent with Bunt et al.’s observations on the need for explanation (Bunt, Lount, and Lauzon 2012). The last was to select programs that were not too complex, where the LLMs’ explanations may have become unsound through hallucination (Kulesza et al. 2015).

We curated a set of eight COBOL programs from the open-source accompaniment of Coughlan (2014)’s introductory COBOL course². We supplemented this set with a Fibonacci number generator³, given its popularity in first-year CS courses. In a group decision-making exercise, five professional software engineers from IBM collaboratively reviewed and eliminated six programs that were either too easily understood, too difficult to understand, or did not possess enough technical depth.

We obfuscated the remaining three programs to remove superficial cues that could make their purpose easily inferable by the LLMs. Specifically, we renamed program identifiers to ‘MyProgram,’ renamed variables, renamed program and method identifiers, and removed code comments. Table 2 shows the three COBOL programs: the Fibonacci sequence generator, a program containing an infinite loop (to see if LLMs could detect the logic bug), and a rock-paper-scissors simulator⁴.

LLM Selection & Explanation Generation We conducted this experiment between June and September 2024, selecting open-weight LLMs (to promote replicability) that were approved for use on our institution’s AI inference platform (IBM’s Big AI Models). Six models fit these criteria:

- LLaMa-3-70b-instruct (LLaMa 3)
- codellama-34b-instruct (CodeLLaMa)
- IBM-LLaMa-2-70b-chat-q (IBM LLaMa 2 Q)
- Granite-13b-Chat-v2 (Granite Chat)

²<https://github.com/Apress/beg-cobol-for-programmers/tree/master>

³<http://progopedia.com/example/fibonacci/341/>

⁴Our obfuscation efforts were successful. The LLMs never identified the Rock-Paper-Scissors game or found the logic bug.

Fibonacci Sequence

```

1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. MyProgram.
3 DATA DIVISION.
4 WORKING-STORAGE SECTION.
5 77 F1 pic 999
6 77 F2 pic 999
7 77 F3 pic 999
8 77 I pic 99
9 77 FST pic XXX
10 77 RES pic X(64)
11 PROCEDURE DIVISION
12 MOVE 0 to I
13 MOVE 0 to F1
14 MOVE 1 to F2
15 MOVE "" to RES
16 perform until I greater than 15
17     ADD F1 to F2 giving F3
18     MOVE F2 to F1
19     MOVE F3 to F2
20     MOVE F1 to FST
21     STRING RES DELIMITED BY SPACE
22     FST DELIMITED BY SIZE
23     ", " DELIMITED BY SIZE INTO
24     RES
25     ADD 1 to I
26 END-PERFORM
27 DISPLAY RES "..."
28 STOP RUN

```

Infinite Loop

```

1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. MyProgram.
3 DATA DIVISION.
4 WORKING-STORAGE SECTION.
5 01 Counters.
6 02 C1 PIC 99.
7 02 C2 PIC 99.
8 02 C3 PIC 9.
9 PROCEDURE DIVISION.
10 Begin.
11 PERFORM Loop VARYING C1
12 FROM 13 BY -5 UNTIL C1
13 LESS THAN 2
14 AFTER C2 FROM 15 BY -4
15 UNTIL C2 LESS THAN 1
16 AFTER C3 FROM 1 BY 1
17 UNTIL C3 GREATER THAN 5
18 STOP RUN.
19 Loop.
20 DISPLAY "Counters 1, 2 and
21 3 are -> "
22 C1 SPACE C2 SPACE C3.

```

Rock-Paper-Scissors

```

1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. MyProgram.
3 DATA DIVISION.
4 WORKING-STORAGE SECTION.
5 01 PL-A PIC 9 VALUE 1.
6 88 R-A VALUE 1.
7 88 P-A VALUE 2.
8 88 S-A VALUE 3.
9 01 PL-B PIC 9 VALUE 2.
10 88 R-B VALUE 1.
11 88 P-B VALUE 2.
12 88 S-B VALUE 3.
13 PROCEDURE DIVISION.
14 BEGIN.
15 DISPLAY "Enter PL-A: " WITH NO ADVANCING
16 ACCEPT PL-A
17 DISPLAY "Enter PL-B: " WITH NO ADVANCING
18 ACCEPT PL-B
19 EVALUATE TRUE ALSO TRUE
20 WHEN R-A ALSO R-B DISPLAY "N/A"
21 WHEN R-A ALSO P-B DISPLAY "PL-B"
22 WHEN R-A ALSO S-B DISPLAY "PL-A"
23 WHEN P-A ALSO R-B DISPLAY "PL-A"
24 WHEN P-A ALSO P-B DISPLAY "N/A"
25 WHEN P-A ALSO S-B DISPLAY "PL-B"
26 WHEN S-A ALSO R-B DISPLAY "PL-B"
27 WHEN S-A ALSO P-B DISPLAY "PL-A"
28 WHEN S-A ALSO S-B DISPLAY "N/A"
29 WHEN OTHER DISPLAY "Error"
30 END-EVALUATE
31 STOP RUN.

```

Table 2: The three COBOL programs used in our work. To see whether LLMs could discern the code, we removed superficial cues (e.g., variable names, program names). In total, we generated 1,072 explanations from these three programs.

- Granite-34b-code-instruct (Granite Code Instruct)
- Mixtral-8x7b-instruct-v01 (Mixtral Instruct)

We performed zero-shot prompting (Kojima et al. 2022) on these LLMs using the 11 prompts we developed. We chose zero-shot prompting to reduce development overhead and preserve adaptation validity.

We generated 1,072 code explanations across all prompts, and we set only three parameters. Two of the parameters governed the number of new tokens generated, which we set to be between 10–1,024 tokens. Our reasoning was that this wide range would enable LLMs to produce both succinct and verbose explanations. The last parameter we set was the temperature, setting it to 0.5 to balance output consistency with response variability⁵. We held these parameters constant throughout our investigation.

Analysis Methods Our investigation focuses solely on whether LLMs made linguistic adaptations in code explanations (rather than explanation accuracy), so we first must define what we mean by “adaptation”.

Definition (Adaptation). Let P be a linguistic phrase of any length. We qualify P as an adaptation for a problem-solving style value if either:

1. P was significantly more likely to occur in one problem-solving style value’s explanations than it was in the opposing value **and** in the control.

⁵As $t \rightarrow 0$, models become more deterministic across independent trials. As $t \rightarrow 1$, models become more stochastic.

2. P occurred **exclusively** in one problem-solving style value and appeared in **at least** 10% of those explanations.

After cleaning the explanations (e.g., lowercasing, removing punctuation), we used scikit-learn’s CountVectorizer to produce n -grams ($n \in [3, 15]$). To remove redundant n -grams, we established two exclusion criteria: 1) the tokens in the n -gram had to appear in the same order in an $(n+1)$ -gram 2) the number of n -gram instances was equal to the number of $(n+1)$ -gram instances. This process left 442,216 n -grams.

To satisfy both adaptation definition conditions, we computed the odds of each n -gram occurring in each problem-solving style value, forming a distribution of odds. To meet our first condition, we excluded n -grams where the odds fell below the midpoint of the distribution, leaving “more likely than not” n -grams. For the second condition, we looked at all n -grams that occurred in *only one* problem-solving style value and excluded the n -grams that did not appear in at least 10% of our data. This yielded 1,029 n -grams.

Three authors collaboratively clustered the n -grams by conceptual similarity, which were independently validated by two non-author researchers. Disagreements were resolved during collaborative discussions by changing a n -gram’s cluster or creating a new cluster until consensus was reached. Then, regular expressions were made for each category. These regular expressions were applied to all code explanations, counting the presence of each category.

Adaptation Category	L	S	R	M	I
1. Explain Step-by-Step	↑	↑			
2. Invite Follow-Up Questions	↑	↑			↑
3. Infer User's Interest	↑		↑	↑	
4. Ask User's Opinion	↑				
5. Normalize Knowledge Gaps		↑			
6. Express Hope of Clarity		↑			↑
7. Hedge Response Language		↓		↓	
8. (Re)Assure the User			↑		
9. Identify Possible Risks			↑		
10. Address Code Complexity			↑		
11. Give Code Purpose Overview	↓			↓	
12. Give Concise Explanations					↑
13. Summarize the Code					↑
Totals:	5	5	4	3	4

Table 3: The 13 adaptations (rows). Each cell’s color shows the problem-solving style value (e.g., **process-oriented**) the LLMs increased (↑) or decreased (↓) the frequency for, relative to its opposing value (e.g., **tinkering-oriented**) and the control. L: Learning Style, S: Self-Efficacy, R: Risk Attitudes, M: Motivations, I: Information Processing Style.

4 Results

How Did These LLMs Adapt? We start with an overview answer for our research question. Table 3 shows a taxonomy of 13 linguistic adaptations (rows) that LLMs made in their code explanations, along with their associated problem-solving style types (columns L–I). Each cell reflects whether that adaptation was more (↑) or less (↓) likely to occur, and the cell colors reflect the adaptations’ associated problem-solving style value. For instance, a ↑ in the L column represents an increased likelihood for **process-oriented** learning style. Similarly, a ↓ in the M column represents a decreased likelihood for **tech-oriented** motivations.

The LLMs achieved generally good coverage for these 10 problem-solving style values. They had at least one adaptation in 8 out of the 10 values, with only **higher** self-efficacy and **task-oriented** motivations adaptations missing. This points to the efficacy of the 10 prompts in generating problem-solving style value adaptations.

Learning Style Adaptations Table 4 (rows) details the first four adaptations in Table 3. For each row, the first three columns show the proportions of unadapted ($\hat{\pi}_u$), **process-oriented** adapted ($\hat{\pi}_p$), and **tinkering-oriented** adapted ($\hat{\pi}_t$) explanations with that adaptation. For example, 30% of the unadapted explanations had *Explain Step-by-Step* instances, compared to 56% of the process-oriented adapted explanations. These presence/absence outcomes naturally lent themselves to odds ratio analyses⁶.

In the *Explain Step-by-Step* adaptation, the LLM stated that it would structure its explanation in methodical ways, like in numeric steps or by division:

Mixtral Instruct (**process-oriented**): “Here’s a step-by-step

⁶Appendix A provides a short primer on proportions ($\hat{\pi}$), odds ($\hat{\omega}$), and odds ratios ($\hat{\phi}$).

explanation of the process:

1. The program starts at the *Begin* section, where the program...

2. Counter C1 starts at 13 and decreases by 5 each time...”

LLaMa 3 (**process-oriented**): “Let’s break down the code step by step:

DATA DIVISION This section defines the variables...

PROCEDURE DIVISION This section contains...”

Table 4’s remaining nine columns show the three odds ratio (OR) statistical tests. Each test shows how much more likely (↑) these adaptations were to occur, including significance (p) and effect size (d)⁷. For instance, the **Explain Step-by-Step** adaptation was almost 3 times more likely (2.97↑) to occur in **process-oriented** explanations than in the unadapted explanations. The adaptation was even more likely (3.49↑) to occur in process-oriented explanations than in tinkering-oriented ones. Both likelihood differences were statistically significant ($p < .001$) with medium effect sizes, with Cohen’s $d \in [0.5, 0.8]$ ⁸.

Two additional sources of evidence triangulate why the *Explain Step-by-Step* adaptation was a **process-oriented** learning style adaptation, the first source being the LLMs themselves. Almost 20% of explanations with this adaptation stated that they took this methodical approach *because* of the user’s learning style:

Mixtral Instruct (**process-oriented**): “[The explanation is] tailored for a process-oriented learner, with clear instructions and step by step explanations.”

The second evidence source comes from prior GenderMag literature. While using GenderMag’s walkthrough, Chatterjee et al. (2024)’s most frequent **process-oriented** issue was “a lack of step-by-step guidance”, which blocked such learners from making progress in online courseware. This points to why the *Explain Step-by-Step* adaptation was appropriate for process-oriented learners.

In Table 4’s second adaptation, *Invite Follow-Up Questions*, the LLMs invited further engagement from the user:

IBM LLaMa 2 Q (**tinkering-oriented**): “Do you have any specific questions about the program...”

CodeLLaMa (**tinkering-oriented**): “If you have any further questions, feel free to ask.”

The data showed that the *Invite Follow-Up Questions* adaptation (Table 4) was geared towards **tinkering-oriented** learners; it was almost twice as likely to occur in tinkering-oriented explanations than in **process-oriented** ones (1.97 ↑), and more than three times more likely than in unadapted (3.30 ↑). Both differences were statistically significant ($p < .05$) with small-to-medium effect sizes.

Over on third of the *Invite Follow-Up Questions* instances (38.5%) revealed this adaptation’s connection to the **tinkering-oriented** learning style. These *Invite Follow-Up Questions* instances invited the tinkering-oriented users to

⁷Haddock et al. (1998) found that $\ln(\hat{\phi}) \left(\sqrt{3/\pi} \right) \approx d$.

⁸We consider Cohen’s $d < .02$ no effect, $d \in [0.2, 0.5]$ a small effect, $d \in [0.5, 0.8]$ a medium effect, and $d \geq 0.8$ a large effect (Cohen 2013).

Adaptation Category	Unad.	Proc.	Tink.	Proc. vs. Unad.			Tink. vs. Unad.			Proc. vs. Tink.		
	$\hat{\pi}_u$	$\hat{\pi}_p$	$\hat{\pi}_t$	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d
Explain Step-by-Step	.30	.56	.26	2.97 ↑	< .001	0.60	≈ 1.00	.365	0.09	3.49 ↑	< .001	0.69
Invite Follow-Up Questions	.10	.16	.27	1.73 ↑	.140	0.30	3.30 ↑	.002	0.66	1.97 ↑	.049	0.35
Infer User's Interest	.00	.00	.22	—	—	—	—	—	—	—	—	—
Ask User's Opinion	.00	.00	.11	—	—	—	—	—	—	—	—	—

Table 4: Statistical details of the first four adaptations. *Unad.*: unadapted explanations. *Proc.*: explanations **process-oriented** adaptations. *Tink.*: **tinkering-oriented** adaptations. **Cols 2–4**: The proportions of unadapted ($\hat{\pi}_u$), process-oriented ($\hat{\pi}_p$) adapted, and tinkering-oriented ($\hat{\pi}_t$) adapted explanations with each adaptation. **Cols 5–13**: The three odds ratio (OR) comparisons, showing how much more (↑) or less (↓) likely an adaptation was, its significance (p), and effect size (Cohen’s d). Dashes: undefined comparison (divide by 0).

engage, asking if they wanted to know how the code worked or how to experiment with it:

LLaMa 3 (**tinkering-oriented**): “Do you have any questions about **how [the code] works** or how you could modify it?”

LLaMa 3 (**tinkering-oriented**): “Do you have any questions about how [the code] works, or would you like to **experiment with** modifying it to see how it behaves?”

This language mirrored the language in two statements in the **tinkering-oriented** prompt, connecting this adaptation to this learning style:

*In order to learn new technology, I tinker with it, constructing my own understanding of **how it works**.*

*My first step in learning new technology is **experimenting** and tinkering with it.*

The STEM education literature provides another source of evidence to explain why the *Invite Follow-Up Questions* adaptation was for **tinkering-oriented** learners. In particular, Vossoughi et al. (2013) suggest that offering suggestions or learning about someone’s ideas, questions, and goals to support the development of engineering projects lies at the heart of tinkering, a position supported by Petrich, Wilkinson, and Bevan (2013). The *Invite Follow-Up Questions* adaptation appears to lean into that pedagogical approach to learn about tinkering-oriented learners’ questions or goals.

The last two adaptations from Table 4 illustrate the second kind of adaptation from Section 3, occurring *exclusively* in **tinkering-oriented** explanations. The *Infer User’s Interest* occurred in 22.1% of tinkering-oriented explanations, where the model said something relating to the user’s interests. The most frequent example of these occurred in the opening sentence of 85.7% explanations, where the LLM stated that the tinkering-oriented user was a COBOL enthusiast:

LLaMa 3 (**tinkering-oriented**): “A *COBOL* enthusiast, eh?”

Despite its prevalence, this adaptation was a surprise for two reasons. First, neither the **tinkering-oriented** prompt nor the existing learning style literature supported a relationship between **tinkering-oriented** learning style and COBOL enthusiasm. Second, every instance of this adaptation came from LLaMa 3. We remain open to the possibility that LLaMa 3’s training data may have found this relationship.

The second adaptation that occurred exclusively for **tinkering-oriented** learners was the *Ask User’s Opinion* adaptation, where the LLMs asked an open-ended, reflective question to the user:

LLaMa 3 (**tinkering-oriented**): “Now I’m curious, what do you think about this program?”

LLaMa 3 (**tinkering-oriented**): “...what do you think this program is intended to do?”

Prior tinkering-related works suggest why the *Ask User’s Opinion* adaptation was a **tinkering-oriented** one. In particular, the last of Dron’s (2014) “10 principles for effective tinkering” calls for tinkerers to reflect while tinkering, telling stories about their reflection. The open-ended structure of the *Ask User’s Opinion* adaptation, which rely on answers beyond binary responses, may be one such method to encourage reflective tinkering.

Self-Efficacy Adaptations Table 5 shows the adaptations from rows 5–7 in Table 3. This table shows the proportions of unadapted ($\hat{\pi}_u$), **lower** self-efficacy ($\hat{\pi}_l$), and **higher** self-efficacy ($\hat{\pi}_h$) explanations.

The first of these adaptations was *Normalize Knowledge Gaps*. In this adaptation, the LLM addressed any potential knowledge gaps, with most instances (85.7%) normalizing the user’s knowledge gap in COBOL:

LLaMa 3 (**lower**): “Don’t worry if you’re not familiar with *COBOL*.”

In these data, the *Normalize Knowledge Gaps* adaptation had the largest likelihood difference between two treatments. It was 35 times more likely to occur in **lower** self-efficacy explanations than in **higher** ones, and it *never* appeared in the unadapted explanations. It was statistically significant, with the largest effect size ($d : 1.97$).

Normalize Knowledge Gaps instances came in three flavors. The first stopped at normalizing the knowledge gap:

LLaMa 3 (**lower**): “Don’t worry if you’re not familiar with the Fibonacci sequence”

The second flavor also explained *why* a knowledge gap may exist. For instance, the LLMs sometimes pointed to COBOL’s age:

LLaMa 3 (**lower**): “Don’t worry...familiar with *COBOL* - it’s an older programming language and it can be a bit tricky...”

LLaMa 3 (**lower**): “Don’t worry...familiar with *cobol* - it’s an older programming language and it can be a bit challenging...”

With the last flavor, the LLMs told the user that they were *a solution* to that knowledge gap, that they would help:

Adaptation Category	Unad.	Low	High	Low vs. Unad.			High vs. Unad.			Low vs. High		
	$\hat{\pi}_u$	$\hat{\pi}_l$	$\hat{\pi}_h$	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d
Normalize Knowledge Gaps	.00	.28	.01	—	—	—	—	—	—	35.67 ↑	< .001	1.97
Express Hope of Clarity	.13	.47	.24	5.73 ↑	< .001	0.96	2.05 ↑	.041	0.40	2.79 ↑	< .001	0.57
Hedge Response Language	.16	.01	.15	18.73 ↓	< .001	1.61	≈ 1	.446	0.07	16.38 ↓	< .001	1.54

Table 5: Statistical summary of the self-efficacy adaptations, in the same format as Table 4. *Unad.*: unadapted explanations. **Low**: lower self-efficacy. **High**: higher self-efficacy. The *Normalize Knowledge Gaps* adaptation was the strongest adaptation, with most instances normalizing a gap in COBOL knowledge.

LLaMa 3 (**lower**): “...not familiar with cobol - I’m here to help you understand what this code does.”

LLaMa 3 (**lower**): “...this code looks unfamiliar - I’m here to help you understand it”

Normalizing knowledge gaps can promote psychological safety, which helps people learn and perform better at their tasks (Edmondson 1999). Additionally Jones et al. (2024) found a positive correlation between psychological safety and help-seeking, which Huet, Motak, and Sakdavong (2016) found that **lower** self-efficacy people were more likely to do. As such, these speak to the appropriateness of the *Normalize Knowledge Gaps* adaptation’s prevalence in lower self-efficacy explanations.

The second adaptation from Table 5 was to *Express Hope of Clarity*. In this adaptation, the LLM expressed that it hoped the explanation had served its the user:

IBM LLaMa 2 Q (**lower**): “I hope this helps you understand what the code does.”

Mixtral Instruct (**lower**): “I hope this explanation helps you understand the code.”

The *Express Hope of Clarity* adaptation was the first linguistic adaptation that adapted for one of the problem-solving style types. Illustrating this, the adaptation was significantly more likely to occur in both **lower** self-efficacy (5.73↑) and **higher** self-efficacy explanations (2.05↑) than they were in unadapted ones. However, we still classify this as a **lower** self-efficacy adaptation because it was significantly more likely to occur in lower than in higher self-efficacy explanations (2.79↑), with a medium effect size.

The *Express Hope of Clarity* adaptation’s prevalence at the self-efficacy problem-solving style type level may have come from the statements in the self-efficacy prompt. In particular, three statements pertained to accessing help:

I am able to use unfamiliar technology when...

...I can call someone for help if I get stuck

...no one is around to help if I need it.

...I have just the built-in help for assistance.

One possibility is that for the LLMs, it did not matter whether someone disagreed (**lower**) or agreed (**higher**) with these statements; they still wanted to make sure that everyone received the help they needed.

All adaptations so far have *increased* linguistic frequency, but the *Hedge Response Language* was the first to *decrease* this frequency. In our data, *Hedge Response Language* occurred when the LLMs hedged their explanations, instead of using definitive language:

Granite Code Instruct (**lower**): “Overall, this cobol code **appears to be** a simple program...”

As Table 5 shows, the LLMs adapted how frequently they hedged their language in **lower** self-efficacy explanations. They were over 18 times *less* likely to hedge their language in lower self-efficacy explanations than in unadapted ones (18.73↓). Similarly, they were over 16 times less likely to hedge their language in lower self-efficacy explanations than in **higher** self-efficacy ones (16.38↓). Both differences were statistically significant with large effect sizes.

Hedging language communicates confidence in a statement (Hyland 1996; Coates 1987), and one explanation may be the LLMs adapting to the prompt. The LLMs were told that **lower** self-efficacy users were more likely to agree with: I am not confident about my ability to use and learn technology. I have other strengths.

Prior works have shown that signaling uncertainty in LLM outputs can reduce over-reliance on those outputs, helping people feel more confident in their final answer (Kim et al. 2024). Increased confidence can result in an increased likelihood that someone will persevere, a reported struggle for those with lower self-efficacy (Bandura and Wessels 1994).

Risk Attitude Adaptations Table 6 shows the three adaptations from rows 8–10 in Table 3. This table shows the proportions of unadapted ($\hat{\pi}_u$), **risk-averse** ($\hat{\pi}_a$), and **risk-tolerant** ($\hat{\pi}_t$) explanations.

The first of these adaptations was *(Re)Assure the User*. This adaptation occurred when the LLM provided assurances (or reassurances) to the user:

Granite Chat (**risk-averse**): “However, I can assure you that...”

In these data, *every* instance of the *(Re)Assure the User* adaptation occurred in **risk-averse** explanations, providing compelling evidence that this adaptation was a risk-averse one. Two additional sources of evidence also point to this conclusion. The first source was that 95% of the explanations with this adaptation mentioned that the assurances were because of the user’s risk-averse attitude:

LLaMa 3 (**risk-averse**): “Given your risk-averse attitude i want to assure you...”

The LLMs occasionally alluded to statements from the **risk-averse** prompt. For example, the LLMs were told that risk-averse people were more likely to agree with:

“I avoid using new apps or technology before they are **well-tested**.”

10% of these explanations mirrored this language while providing reassurances:

Adaptation Category	Unad.	Ave.	Tol.	Ave. vs. Unad.			Tol. vs. Unad.			Ave. vs. Tol.		
	$\hat{\pi}_u$	$\hat{\pi}_a$	$\hat{\pi}_t$	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d
(Re)Assure the User	.00	.19	.00	—	—	—	—	—	—	—	—	—
Identify Possible Risks	.00	.22	.01	—	—	—	—	—	—	27.68 ↑	< .001	1.83
Address Code Complexity	.01	.22	.11	5.27 ↑	< .001	0.92	2.08	.146	0.404	1.97 ↑	.076	0.40

Table 6: Statistical summary of the risk attitudes adaptations, in the same format as Table 5. *Unad.*: unadapted explanations. *Ave.*: risk-averse. *Tol.*: risk-tolerant. These adaptations targeted potential costs and risks for the user, including complexity.

Adaptation Category	Unad.	Task	Tech	Task vs. Unad.			Tech vs. Unad.			Task vs. Tech		
	$\hat{\pi}_u$	$\hat{\pi}_{ta}$	$\hat{\pi}_{te}$	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d
Give Code Purpose Overview	.57	.54	.33	≈ 1	.393	0.07	2.74 ↓	< .001	0.55	2.43 ↓	.002	0.49

Table 7: Statistical summary of the motivations adaptations, in the same format as Table 6. *Unad.*: unadapted explanations. *Task*: task-oriented. *Tech*: tech-oriented. This adaptation may have been the LLMs’ attempt to keep the learning “fun”.

LLaMa 3 (risk-averse): “...this code doesn’t involve any complex or untested features”

The second adaptation from Table 6 was **Identify Possible Risks**, where the LLMs addressed whether the user would face any risks with the code:

Mixtral Instruct (risk-averse): “The code is easy to understand and test, making it less likely to cause any unexpected issues or risks.”

Mixtral Instruct (risk-averse): “[The code] does not use any advanced features or take any risks with your data and it provides clear feedback based on the input you provide.”

The statistical evidence suggested that this was a risk-averse adaptation. Illustrating this, this adaptation was almost 28 times more likely to occur in risk-averse explanations than in risk-tolerant ones (27.68↑). This likelihood difference was significant, with a large effect size.

Prior work have investigated how to generate LLM risk identification processes (Adejumo and Johnson 2025), and prior risk attitude literature helps explain the importance of the (Re)Assure the User and Identify Possible Risks adaptations for risk-averse users. When it comes to cost/benefit analyses, people with more risk-averse attitudes place a greater emphasis on the perceived costs (e.g., Kind, Wouter Botzen, and Aerts, 2017; Kramer et al., 2025; Li and Shi, 2026), selecting safer actions when the outcomes are uncertain. Proactive assurances of possible risks could become effective elucidation tools for uncertain outcomes, particularly when using unfamiliar technology.

The last adaptation in Table 6 was **Address Code Complexity**, where the LLM made observations about the code:

Mixtral Instruct (risk-averse): “This is a simple, straightforward COBOL program...”

LLaMa 3 (risk-averse): “[The code] doesn’t involve any complex logic or external dependencies”

The statistical evidence provided evidence to suggest this being a risk-averse adaptation. This adaptation was over five times more likely to appear in risk-averse explanations than in unadapted explanations (5.27 ↑), a significant difference with a large effect size. However, despite it being

almost twice as likely to occur in risk-averse explanations than in risk-tolerant ones (1.97 ↑), this difference was only marginally significant ($p \in [0.05, 0.1)$), with a small effect size. Nonetheless, the directional consistency across both comparisons supports classifying this adaptation as a risk-averse adaptation.

Motivations Adaptations Table 7 shows the 11th adaptation from Table 3. This table shows the proportions of unadapted ($\hat{\pi}_u$), task-oriented ($\hat{\pi}_{ta}$), and tech-oriented ($\hat{\pi}_{te}$) explanations. The *Give Code Purpose Overview* occurred when the LLM provided a high-level overview of the code’s purpose:

Granite Code Instruct (task-oriented): “The provided cobol code is a program that calculates and displays the Fibonacci...”

This was another adaptation with reduced frequency for one of the problem-solving style values, namely tech-oriented motivations. The LLMs were significantly less likely to provide an overview in tech-oriented explanations than unadapted (2.74↓) or task-oriented ones (2.43↓).

Liquin and Lombrozo’s (2020) work on explanation-seeking behaviors may help explain why this was a tech-oriented explanation. In particular, their *Anticipated Learning and Information Gap* triggers. Those with tech-oriented motivations view technology as a source of fun, where they are more likely to agree with statements like:

I make time to explore technology that is not critical to my job.

I spend time and money on technology just because it’s fun.

One possible explanation may be that code purpose overviews could suppress one or more of these triggers, where the fun of learning new material (anticipated learning) may be nullified, possibly by closing the information gap “too early”. Thus, a reduced frequency of the *Give Code Purpose Overview* adaptation may keep curiosity alive for those with tech-oriented motivations.

Information Processing Style Adaptations Table 8 shows the last two adaptations from Table 3. This table shows the proportions of unadapted, comprehensive ($\hat{\pi}_c$), and selective ($\hat{\pi}_s$) information processing explanations.

Adaptation Category	Unad.	Comp.	Sele.	Comp. vs. Unad.			Sele. vs. Unad.			Comp. vs. Sele.		
	$\hat{\pi}_u$	$\hat{\pi}_c$	$\hat{\pi}_s$	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d	$\hat{\phi}$	p	d
Give Concise Explanations	.00	.00	.23	—	—	—	—	—	—	—	—	—
Summarize the Code	.44	.57	.38	1.69 ↑	.044	0.29	≈ 1	.272	0.12	2.10 ↑	.007	0.41

Table 8: Statistical summary of the motivations adaptations, in the same format as Table 7. *Unad.*: unadapted explanations. *Comp.*: comprehensive. *Sele.*: selective. These focused on how much information to give (top) and where to give it (bottom).

The *Give Concise Explanations* adaptation when the LLM stated that it had given a concise explanation:

LLaMa 3 (selective): “...I’ve tried to provide a concise and to-the-point explanation.”

This adaptation occurred only in selective information processing style explanations, and the LLMs confirmed that the explanations were concise *because* of this problem-solving style value:

LLaMa 3 (selective): “...more selective information processing style, you might appreciate this concise explanation of the code.”

The LLMs stated that they had given a concise explanation, which was a verifiable claim. Almost all of the explanations with *Give Concise Explanations* instances came from LLaMa 3 (95.5%), so we analyzed every unadapted, comprehensive, and selective explanation, using word count as a measure of conciseness.

Figure 1 shows the word distribution for these three treatments, with no visible differences between the unadapted (top) and selective (bottom) explanations. However, our statistical findings *did* uncover a significant difference with a large effect size (ANOVA, $F(2,122)$: 16.90, p : < .0001, η^2 : .277)⁹, just not in the way we expected. When we removed the comprehensive explanations, we also removed the significance (ANOVA, $F(1,81)$: 0.054, p : .817, η^2 : < .01). This meant that LLaMa 3 used significantly more words on average in comprehensive explanations than it did in either the unadapted or selective explanations.

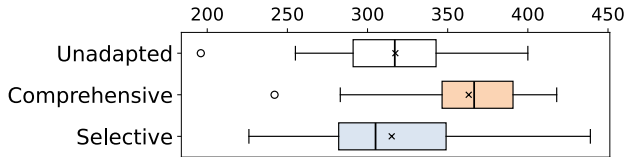


Figure 1: Number of words (x-axis) for unadapted, comprehensive, and selective adapted code explanations. LLaMa 3 used significantly more words on average for comprehensive adapted explanations.

These results point to a personalization challenge. Although LLaMa 3 stated how it *should* have adapted, it did not deliver in this case. After extracting these linguistic adaptations, the challenges become how to verify if an LLM delivers *and* how to steer them effectively if they do not.

⁹We consider $\eta^2 \in [0, .01)$ no effect, $\eta^2 \in [.01, .06)$ a small effect, $\eta^2 \in [.06, .14)$ a medium effect, and $\eta^2 \geq .14$ a large effect.

The *Summarize the Code* adaptation occurred when the LLM summarized the code at the *end* of the explanation:

Granite Chat (comprehensive): “In summary, this COBOL code is a simple decision-making...”

Granite Code Instruct (comprehensive): “In summary, this program generates the first 15 numbers in the Fibonacci sequence...”

The statistical evidence suggested that this adaptation was for comprehensive information processors. The LLMs were 1.69 times more likely to summarize the code in comprehensive explanations than in unadapted ones, and they were over twice as likely than in selective ones (2.10↑). Both differences were significant, with small-to-medium effect sizes.

Pedagogically, this adaptation made sense, since explanations are educational tools (Williams, Lombrozo, and Rehder 2013). In educational frameworks, explanation summaries aim to review and summarize what students learned, reinforcing the learned material (Hunter 1994). The *Summarize the Code* adaptation appeared towards the end of an explanation, potentially reinforcing the explanations for those who prefer more information while problem-solving.

How Frequently Did These LLMs Adapt? Throughout Section 4, we showed how these six LLMs adapted to GenderMag’s 10 problem-solving style values. However, which LLM (if any) adapted most frequently?

Figure 2 directly answers this question, showing how frequently each LLM generated adaptations per explanation. As the figure shows, CodeLLaMa produced approximately 1 adaptation per explanation, the fewest of these LLMs. However, LLaMa 3 produced about 4 adaptations per explanation on average, which was significantly more than even the second most-frequent adaptation generator, Mixtral Instruct (one-sided t-test, $t(553)$: 14.871, p : < .0001, d : 1.62). This is consistent with other LLM personalization literature, where multiple investigations have found LLaMa 3 as the most capable of adapting its responses to various roles/personas (Pal, Das, and Srihari 2024), even showing comparable results to LLMs like GPT-4.1 (Samuel et al. 2024).

5 Discussion: Implications for Design

The taxonomy in this paper demonstrates how LLMs linguistically adapted their code explanations to align with the needs of diverse problem-solving style values, but one challenge for human-AI interaction practitioners is that the GenderMag problem-solving style value survey may not be a scalable solution to manifest these adaptations in LLM-generated code explanations. In total, the survey has 36 questions, so its length may present a barrier for users.

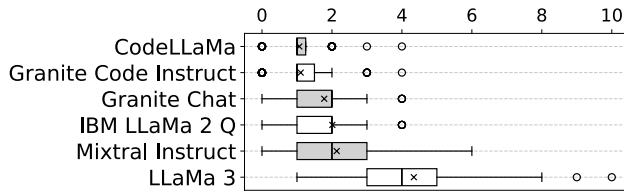


Figure 2: The number of adaptations per explanation, by LLM. On average, LLaMa 3 generated significantly more adaptations than even the second most-frequent generator.

Figure 3 illustrates an approach inspired by Brachman et al. (2025) to manifest these adaptations via explicit UI controls. This design empowers users with agency over whether specific adaptations appear in their code explanations. It also provides a starting point for users to explore what other adaptations they may wish to see, a known challenge in generative AI (Zamfirescu-Pereira et al. 2023; Dang et al. 2022). One potential limitation is the potential for choice overload (Iyengar and Lepper 2000); the 13 adaptations in this paper, with binary options, would mean $2^{13} = 8,192$ possible configurations for the user.



Figure 3: UI controls for three adaptations in Table 3.

One solution could be grouping adaptations by problem-solving style value. Figure 4 illustrates a design for users to select which value they most align with on a slider (Figure 4, left), for all five problem-solving style types. With these selections, the models could incorporate the group of relevant adaptations into their code explanations. For example, if someone aligns more with a **risk-averse** attitude, as in the figure, then the LLM could incorporate the three risk attitude adaptations from Table 3. To maintain control over the responses, users could still adjust these options to suit their needs (Figure 4, bottom-right). This design accomplishes the higher level grouping and maintains the user’s agency. Although this design reduces the chance of choice overload, it may pose memory and inconsistent appearance challenges in the UI, known design “traps” (Medlock and Herbst 2020).

6 Concluding Remarks & Future Work

This paper is the first to investigate how Large Language Models (LLMs) adapt their code explanations to align with the needs of diverse problem-solving style values. To accomplish this, we developed 10 prompts from a validated problem-solving style value survey and generated 1,072 COBOL code explanations with six open-source LLMs. We

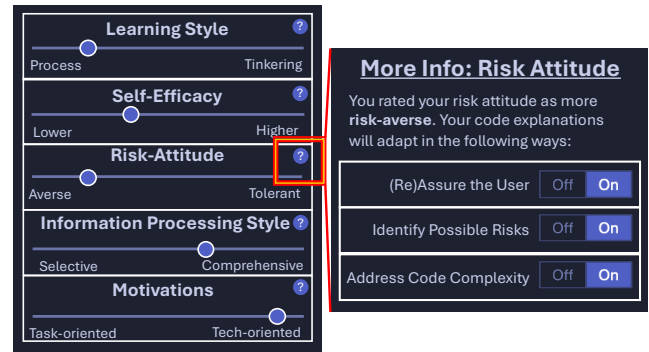


Figure 4: UI controls at higher grouping levels on the five problem-solving style types. **(Left):** For each type, users could select their preferred value. They can access more information on-demand (callout). **(Right):** Users can find the preset adaptation values (i.e., Table 3 cells, where $\uparrow \Rightarrow$ “on”, and $\downarrow \Rightarrow$ “off”).

examined those explanations using natural language processing and statistical techniques, discovering an initial taxonomy 13 linguistic adaptation categories for 8 of the 10 problem-solving style values. Additionally, we identified how many adaptation instance these LLMs made relative to each other, suggesting that model selection remains an important choice for personalized AI code explanations.

This paper is the first to investigate how effectively LLMs adapt to GenderMag’s five problem-solving style types, and we see multiple approaches to build upon this work. One approach to expand the taxonomy could investigate more recently released LLMs, whether open-weight models like LLaMa 4 or GPT-OSS or proprietary ones like Anthropic’s Claude family. A second expansion approach could broaden which programs are explained, including more COBOL programs and programming languages. A third expansion approach could investigate adaptations along different diversity dimensions, like other inclusive design methods (Busteed et al. 2026) or demographic dimensions like the “Big Five” OCEAN personality traits (Goldberg 1992). A fourth expansion approach could move beyond code explanations. One possibility lies in exploring explanation adaptations in domains like medicine and law. Another possibility lies in exploring adaptations beyond explanations, such as in AI planning outcomes. Any of these four approaches could generalize the existing adaptations or discover more, building a more robust taxonomy of LLM adaptations. Additionally, this work could be built upon by empirically investigating whether (and how) these code explanation adaptations support people who have no COBOL development background in code understanding tasks.

When people engage with explanations, they are inherently problem-solving, but explanations are not “one-size-fits-all”. This paper’s taxonomy reflects a first step towards a future where LLM code explanations can meet diverse problem-solvers’ equally diverse problem-solving needs.

References

- Adejumo, E. K.; and Johnson, B. 2025. Explaining Code Risk in OSS: Towards LLM-Generated Fault Prediction Interpretations. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, 199–203.
- Agarwal, V.; Jablokow, K.; and McComb, C. 2025. Putting the Ghost in the Machine: Emulating Cognitive Style in Large Language Models. *Journal of Computing and Information Science in Engineering*, 25(2): 021002.
- Anderson, A.; Dodge, J.; Sadarangani, A.; Juozapaitis, Z.; Newman, E.; Irvine, J.; Chattopadhyay, S.; Fern, A.; and Burnett, M. 2019. Explaining reinforcement learning to mere mortals: an empirical study. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 1328–1334.
- Anderson, A.; Dodge, J.; Sadarangani, A.; Juozapaitis, Z.; Newman, E.; Irvine, J.; Chattopadhyay, S.; Olson, M.; Fern, A.; and Burnett, M. 2020. Mental Models of Mere Mortals with Explanations of Reinforcement Learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 10(2): 1–37.
- Anderson, A.; Guevara, J. N.; Moussaoui, F.; Li, T.; Vorvoreanu, M.; and Burnett, M. 2024. Measuring User Experience Inclusivity in Human-AI Interaction via Five User Problem-Solving Styles. *ACM Transactions on Interactive Intelligent Systems*, 14(3): Paper No. 21, 1–90.
- Anderson, A.; Moussaoui, F.; Noa Guevara, J.; Hamid, M. M.; and Burnett, M. 2026. “Over-the-Hood” AI Inclusivity Bugs and How 3 AI Product Teams Found and Fixed Them. In *Proceedings of the 31st International Conference on Intelligent User Interfaces, UII ’26*, (To Appear). New York, NY, USA: Association for Computing Machinery.
- Anik, A. I.; and Bunt, A. 2024. Supporting User Critiques of AI Systems via Training Dataset Explanations: Investigating Critique Properties and the Impact of Presentation Style. In *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 134–147.
- Arya, V.; Bellamy, R. K. E.; Chen, P.-Y.; Dhurandhar, A.; Hind, M.; Hoffman, S. C.; Houde, S.; Liao, Q. V.; Luss, R.; Mojsilović, A.; Mourad, S.; Pedemonte, P.; Raghavendra, R.; Richards, J. T.; Sattigeri, P.; Shanmugam, K.; Singh, M.; Varshney, K. R.; Wei, D.; and Zhang, Y. 2020. AI Explainability 360: An Extensible Toolkit for Understanding Data and Machine Learning Models. *Journal of Machine Learning Research*, 21(130): 1–6.
- Asthana, S.; Rashkin, H.; Clark, E.; Huot, F.; and Lapata, M. 2024. Evaluating LLMs for Targeted Concept Simplification for Domain-Specific Texts. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 6208–6226. Miami, Florida, USA: Association for Computational Linguistics.
- Bandura, A. 1986. The explanatory and predictive scope of self-efficacy theory. *Journal of social and clinical psychology*, 4(3): 359–373.
- Bandura, A.; and Wessels, S. 1994. *Self-efficacy*, volume 4. na.
- Becker, B. A.; Craig, M.; Denny, P.; Keuning, H.; Kiesler, N.; Leinonen, J.; Luxton-Reilly, A.; Prather, J.; and Quille, K. 2023. Generative ai in introductory programming. *Computer Science Curricula*, 438–439.
- Brachman, M.; Goldberg, A.; Anderson, A.; Houde, S.; Muller, M.; and Weisz, J. D. 2025. Towards Personalized and Contextualized Code Explanations. In *Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*, 120–125.
- Bunt, A.; Lount, M.; and Lauzon, C. 2012. Are explanations always important? A study of deployed, low-cost intelligent interactive systems. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, 169–178.
- Burnett, M.; Stumpf, S.; Macbeth, J.; Makri, S.; Beckwith, L.; Kwan, I.; Peters, A.; and Jernigan, W. 2016. GenderMag: A method for evaluating software’s gender inclusiveness. *Interacting with Computers*, 28(6): 760–787.
- Burnett, M. M.; Beckwith, L.; Wiedenbeck, S.; Fleming, S. D.; Cao, J.; Park, T. H.; Grigoreanu, V.; and Rector, K. 2011. Gender pluralism in problem-solving software. *Interacting with computers*, 23(5): 450–460.
- Busteed, A.; Noa-Guevara, J.; Castro, L. A.; Moz Ruiz, D.; Afroz, S.; Mokraoui, I.; Velhal, P.; Morreale, P.; Sarma, A.; and Burnett, M. 2026. “Fast, easy, simple”? SES-diverse transfer students’ sociotechnical experiences registering for classes. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems, CHI ’26*. New York, NY, USA: Association for Computing Machinery. ISBN 9798400722783.
- Chapagain, J.; Sajib, M. I.; Prodan, R.; and Rus, V. 2024. A Study of LLM Generated Line-by-Line Explanations in the Context of Conversational Program Comprehension Tutoring Systems. In Ferreira Mello, R.; Rummel, N.; Jivet, I.; Pishtari, G.; and Ruipérez Valiente, J. A., eds., *Technology Enhanced Learning for Inclusive and Equitable Quality Education*, 64–74. Cham: Springer Nature Switzerland. ISBN 978-3-031-72315-5.
- Chatterjee, A.; Choudhuri, R.; Sarkar, M.; Chattopadhyay, S.; Liu, D.; Hedao, S.; Burnett, M.; and Sarma, A. 2024. Debugging for Inclusivity in Online CS Courseware: Does it Work? In *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1*, 419–433.
- Chatti, M. A.; Guesmi, M.; Vorgerd, L.; Ngo, T.; Joarder, S.; Ain, Q. U.; and Muslim, A. 2022. Is More Always Better? The Effects of Personal Characteristics and Level of Detail on the Perception of Explanations in a Recommender System. In *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization, UMAP ’22*, 254–264. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392075.
- Chen, C.-Y. 2025. Effects of Worked Examples with Explanation Types and Learner Motivation on Cognitive Load and Programming Problem-Solving Performance. *ACM Trans. Comput. Educ.*, 25(2).

- Coates, J. 1987. Epistemic modality and spoken discourse. *Transactions of the Philological society*, 85(1): 110–131.
- Cohen, J. 2013. *Statistical power analysis for the behavioral sciences*. routledge.
- Coughlan, M. 2014. *Beginning COBOL for Programmers*. Apress.
- Dang, H.; Mecke, L.; Lehmann, F.; Goller, S.; and Buschek, D. 2022. How to prompt? Opportunities and challenges of zero-and few-shot learning for human-AI interaction in creative applications of generative models. *arXiv preprint arXiv:2209.01390*.
- Dodge, J.; Anderson, A.; Khanna, R.; Irvine, J.; Dikkala, R.; Lam, K.-H.; Tabatabai, D.; Ruangrotsakun, A.; Shureih, Z.; Kahng, M.; et al. 2021. From “no clear winner” to an effective Explainable Artificial Intelligence process: An empirical journey. *Applied AI Letters*, 2(4): e36.
- Dodge, J.; Liao, Q. V.; Zhang, Y.; Bellamy, R. K. E.; and Dugan, C. 2019. Explaining Models: An Empirical Study of How Explanations Impact Fairness Judgment. In *Proceedings of the 24th International Conference on Intelligent User Interfaces, IUI '19*, 275–285. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362726.
- Dron, J. 2014. Ten principles for effective tinkering. In *eLearn: World Conference on EdTech*, 505–513. Association for the Advancement of Computing in Education (AACE).
- Edmondson, A. 1999. Psychological Safety and Learning Behavior in Work Teams. *Administrative Science Quarterly*, 44(2): 350–383.
- Fallatah, A.; Hamid, M. M.; Moussaoui, F. A.; Chikezie, C.; Erwig, M.; Bogart, C.; Sarma, A.; and Burnett, M. 2025. Intersectional HCI on a budget: An analytical approach powered by types. *International Journal of Human-Computer Interaction*, 41(21): 13599–13622.
- Feng, Z.; Guo, D.; Tang, D.; Duan, N.; Feng, X.; Gong, M.; Shou, L.; Qin, B.; Liu, T.; Jiang, D.; et al. 2020. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*.
- Geuenich, J.; Joublin, F.; Ceravola, A.; Brandenburg, S.; and Dodge, J. 2026. AI Can See What You Can’t See: How LLM-Agents Complement Human-Based Gender-Inclusive Usability Testing. *ACM Trans. Interact. Intell. Syst.* Just Accepted.
- Ghammam, A.; Rzig, D. E.; Almukhtar, M.; Khalsi, R.; Hassan, F.; and Kessentini, M. 2025. Build Code Needs Maintenance Too: A Study on Refactoring and Technical Debt in Build Systems. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, 616–628. IEEE.
- Goldberg, L. R. 1992. The development of markers for the Big-Five factor structure. *Psychological Assessment*, 4: 26–42.
- Guo, Y.; Chang, J. C.; Antoniak, M.; Bransom, E.; Cohen, T.; Wang, L.; and August, T. 2024. Personalized Jargon Identification for Enhanced Interdisciplinary Communication. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 4535–4550. Mexico City, Mexico: Association for Computational Linguistics.
- Hadad, R. 2025. The use of makerspaces for the development of computational thinking skills and dispositions: pedagogical practices facilitators use. *Computer Science Education*, 35(4): 754–793.
- Haddock, C. K.; Rindskopf, D.; and Shadish, W. R. 1998. Using odds ratios as effect sizes for meta-analysis of dichotomous data: a primer on methods and issues. *Psychological methods*, 3(3): 339.
- Hamid, M. M.; Chatterjee, A.; Guizani, M.; Anderson, A.; Moussaoui, F.; Yang, S.; Escobar, I.; Sarma, A.; and Burnett, M. 2024. *How to Measure Diversity Actionably*, chapter 27. Apress.
- Hamid, M. M.; Moussaoui, F.; Guevara, J. N.; Anderson, A.; Agarwal, P.; Dodge, J.; and Burnett, M. 2026. Inclusive Design of AI’s Explanations: Just for Those Previously Left Out? *ACM Transactions on Interactive Intelligent Systems*, (1): Paper No. 8, 44 pages.
- Harry, T.; Ngong, I.; Nweke, C.; Feng, Y.; and Near, J. 2026. Beyond Fixed Psychological Personas: State Beats Trait, but Language Models are State-Blind. *arXiv preprint arXiv:2601.15395*.
- Haver, H. L.; Lin, C. T.; Sirajuddin, A.; Yi, P. H.; and Jeudy, J. 2023. Use of ChatGPT, GPT-4, and Bard to Improve Readability of ChatGPT’s Answers to Common Questions About Lung Cancer and Lung Cancer Screening. *American Journal of Roentgenology*, 221(5): 701–704. PMID: 37341179.
- Huet, N.; Motak, L.; and Sakdavong, J. C. 2016. Motivation to seek help and help efficiency in students who failed in an initial task. *Computers in Human Behavior*, 63: 584–593.
- Hunter, M. 1994. *Mastery teaching*. ERIC.
- Hyland, K. 1996. Writing without conviction? Hedging in science research articles. *Applied linguistics*, 17(4): 433–454.
- Iyengar, S. S.; and Lepper, M. R. 2000. When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology*, 79(6): 995.
- Jones, M. S.; Cravens, A. E.; Zarestky, J.; Ngai, C.; and Love, H. B. 2024. Facilitating psychological safety in science and research teams. *Humanities and Social Sciences Communications*, 11(1): 1632.
- Kim, S. S. Y.; Liao, Q. V.; Vorvoreanu, M.; Ballard, S.; and Vaughan, J. W. 2024. “I’m Not Sure, But...”: Examining the Impact of Large Language Models’ Uncertainty Expression on User Reliance and Trust. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, FAccT ’24*, 822–835. New York, NY, USA: Association for Computing Machinery. ISBN 9798400704505.
- Kind, J.; Wouter Botzen, W.; and Aerts, J. C. 2017. Accounting for risk aversion, income distribution and social welfare in cost-benefit analysis for flood risk management. *WIREs Climate Change*, 8(2): e446.

- Kirton, M. J. 1984. Adaptors and innovators—Why new initiatives get blocked. *Long range planning*, 17(2): 137–143.
- Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.
- Kramer, B.; Braun, M.; Ruiz, P. S.; and Taheri, H. 2025. Integrating Gender in Cost-Benefit Analysis of Climate Interventions: Training Guide for the Regional Universities Forum for Capacity Building in Agriculture (RUFORUM). *AICCRA Training Material*.
- Kulesza, T.; Burnett, M.; Wong, W.-K.; and Stumpf, S. 2015. Principles of Explanatory Debugging to Personalize Interactive Machine Learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, 126–137. New York, NY, USA: Association for Computing Machinery. ISBN 9781450333061.
- Lee, A. 2020. Wanted: People who know a half century-old computer language so states can process unemployment claims — CNN business.
- Lei, F.; Liu, J.; Noei, S.; Zou, Y.; Truong, D.; and Alexander, W. 2025. Enhancing COBOL Code Explanations: A Multi-Agents Approach Using Large Language Models. *arXiv preprint arXiv:2507.02182*.
- Leinonen, J.; Denny, P.; MacNeil, S.; Sarsa, S.; Bernstein, S.; Kim, J.; Tran, A.; and Hellas, A. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2023, 124–130. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701382.
- Li, X.; and Shi, B. 2026. The mechanism and optimization of contract matching between new agricultural entities and agricultural land transfer model: A cost-benefit and game analysis. *Land Use Policy*, 167: 108047.
- Liang, J.; Hou, Y.; Zhou, S.; Chen, J.; Xiong, Y.; and Huang, G. 2019. How to Explain a Patch: An Empirical Study of Patch Explanations in Open Source Projects. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 58–69.
- Liang, J. T.; Yang, C.; and Myers, B. A. 2024. A large-scale survey on the usability of ai programming assistants: Successes and challenges. In *Proceedings of the 46th IEEE/ACM international conference on software engineering*, 1–13.
- Liquin, E. G.; and Lombrozo, T. 2020. A functional approach to explanation-seeking curiosity. *Cognitive Psychology*, 119: 101276.
- Luo, Q.; Ye, Y.; Liang, S.; Zhang, Z.; Qin, Y.; Lu, Y.; Wu, Y.; Cong, X.; Lin, Y.; Zhang, Y.; et al. 2024. Repoagent: An llm-powered open-source framework for repository-level code documentation generation. *arXiv preprint arXiv:2402.16667*.
- MacNeil, S.; Tran, A.; Mogil, D.; Bernstein, S.; Ross, E.; and Huang, Z. 2022. Generating diverse code explanations using the gpt-3 large language model. In *Proceedings of the 2022 ACM conference on international computing education research-volume 2*, 37–39.
- Martin, C. R. 2020. Brush up your COBOL: Why is a 60 year old language suddenly in demand? *Stack Overflow Blog*.
- Medlock, M.; and Herbst, S. 2020. Ui Tenets & Traps.
- Mentonelli, G. 2025. Why COBOL still dominates banking and how to modernize.
- Meta. 2023. Meta LLaMa. https://github.com/meta-llama/llama/blob/main/example_chat_completion.py.
- Murgia, E.; Pera, M. S.; Landoni, M.; and Huibers, T. 2023. Children on ChatGPT Readability in an Educational Context: Myth or Opportunity? In *Adjunct Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '23 Adjunct, 311–316. New York, NY, USA: Association for Computing Machinery. ISBN 9781450398916.
- Nam, D.; Macvean, A.; Hellendoorn, V.; Vasilescu, B.; and Myers, B. 2024. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 1–13.
- Oli, P.; Banjade, R.; Chapagain, J.; and Rus, V. 2023. The behavior of large language models when prompted to generate code explanations. *arXiv preprint arXiv:2311.01490*.
- Ouyang, W.; Szewczyk, S. H.; and Miles, J. 2025. Female directors on the sensitivity of investment to growth opportunities. *Accounting & Finance*, 65(1): 713–752.
- Pal, S.; Das, S.; and Srihari, R. K. 2024. Beyond Discrete Personas: Personality Modeling Through Journal Intensive Conversations. *arXiv preprint arXiv:2412.11250*.
- Petrich, M.; Wilkinson, K.; and Bevan, B. 2013. It looks like fun, but are they learning? In *Design, make, play*, 50–70. Routledge.
- Phillips, J.; Bowes, D.; El-Haj, M.; and Hall, T. 2022. Improved Evaluation of Automatic Source Code Summarisation. In Bosselut, A.; Chandu, K.; Dhole, K.; Gangal, V.; Gehrmann, S.; Jernite, Y.; Novikova, J.; and Perez-Beltrachini, L., eds., *Proceedings of the Second Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, 326–335. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics.
- Ramsey, F.; and Schafer, D. 2012. *The statistical sleuth: a course in methods of data analysis*. Cengage Learning.
- Ribeiro, J. V. A.; Correia, T. P.; Requena, J. V. S. C.; and Berton, L. 2026. Evaluating Reference-Free Summarization Quality Metrics for Portuguese: A Study with Human Judgments in Financial News. In Souza, M.; de Dios-Flores, I.; Santos, D.; Freitas, L.; Souza, J. W. d. C.; and Ribeiro, E., eds., *Proceedings of the 17th International Conference on Computational Processing of Portuguese (PROPOR 2026) - Vol. 1*, 899–907. Salvador, Brazil: Association for Computational Linguistics. ISBN 979-8-89176-387-6.
- Rooiein, D.; Curry, A. C.; and Hovy, D. 2023. Know Your Audience: Do LLMs Adapt to Different Age and Education Levels? *arXiv preprint arXiv:2312.02065*.

- Sakdavong, J.-C.; and Peyrègne, M. 2025. How gender and age mediate the effect of self-efficacy on digital learning performance? *Communications in Computer and Information Science*.
- Samuel, V.; Zou, H. P.; Zhou, Y.; Chaudhari, S.; Kalyan, A.; Rajpurohit, T.; Deshpande, A.; Narasimhan, K.; and Mura-hari, V. 2024. Personagym: Evaluating persona agents and llms. *arXiv preprint arXiv:2407.18416*.
- Schäfer, M.; Nadi, S.; Eghbali, A.; and Tip, F. 2023. An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering*, 50(1): 85–105.
- Stumpf, S.; Peters, A.; Bardzell, S.; Burnett, M.; Busse, D.; Cauchard, J.; and Churchill, E. 2020. Gender-Inclusive HCI Research and Design: A Conceptual Review. *Foundations and Trends in Human-Computer Interaction*, 13(1): 1–69.
- Sun, W.; Miao, Y.; Li, Y.; Zhang, H.; Fang, C.; Liu, Y.; Deng, G.; Liu, Y.; and Chen, Z. 2025. Source Code Summarization in the Era of Large Language Models. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, 1882–1894.
- Sundaram, G.; Venkatesh V. B.; and K B, S. 2025. DocStringEval: Evaluating the Effectiveness of Language Models for Code Explanation Through DocString Generation. In *2025 International Conference on Emerging Technologies in Computing and Communication (ETCC)*, 1–7.
- Tan, B. C. Z.; Liu, Z.; Yi, X.; Yao, J.; Xie, X.; Chen, N. F.; and Lee, R. K.-W. 2026. Can Persona-Prompted LLMs Emulate Subgroup Values? An Empirical Analysis of Generalisability and Fairness in Cultural Alignment. *arXiv preprint arXiv:2604.12851*.
- Tsakalakis, N.; Stalla-Bourdillon, S.; Huynh, D.; and Moreau, L. 2025. A typology of explanations to support Explainability-by-Design. *ACM J. Responsib. Comput.*, 2(1).
- Vossoughi, S.; Escudé, M.; Kong, F.; and Hooper, P. 2013. Tinkering, learning & equity in the after-school setting. In *annual FabLearn conference*. Palo Alto, CA: Stanford University.
- Weisz, J. D.; Kumar, S. V.; Muller, M.; Browne, K.-E.; Goldberg, A.; Heintze, K. E.; and Bajpai, S. 2025. Examining the use and impact of an ai code assistant on developer productivity and experience in the enterprise. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 1–13.
- Williams, J. J.; Lombrozo, T.; and Rehder, B. 2013. The hazards of explanation: Overgeneralization in the face of exceptions. *Journal of Experimental Psychology: General*, 142(4): 1006.
- Zamfirescu-Pereira, J. D.; Wong, R. Y.; Hartmann, B.; and Yang, Q. 2023. Why Johnny can’t prompt: how non-AI experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI conference on human factors in computing systems*, 1–21.