Feedback Literacy: Holistic Analysis of Secondary Educators' Views of LLM Explanations of Program Error Messages

Veronica Cucuiat
veronica.cucuiat@raspberrypi.org
Raspberry Pi Foundation
Cambridge, United Kingdom

Jane Waite
jane.waite@raspberrypi.org
Raspberry Pi Foundation
Cambridge, United Kingdom

ABSTRACT

The implications of using large language model (LLM) tools for learning to program at secondary school level are largely unknown, and yet there is pressure for teachers to engage with these. To start addressing this gap, we investigated: *RQ1: What are secondary educators' views on the potential classroom use of LLM program error message explanations? RQ2: In what ways can a feedback literacy perspective support the analysis of educators' views of potential classroom use of LLM program error message explanations?* The responses of eight expert secondary school educators were gathered during a semi-structured, activity-based interview and qualitatively analysed. Fifteen themes were derived from their commentary, of which ten corresponded to enhanced program error message (PEM) guidelines. Yet, all themes correlated to feedback literacy theory, providing a more holistic view. The analysis revealed that educators preferred LLM explanations to *guide* and *develop understanding* rather than *tell*, that students should be supported to *make judgements* and *action* LLM-generated feedback. Combining PEM guideline and feedback literacy findings, we suggest augmented IDEs should be designed with educators and students in mind, and teacher professional development (PD) is needed. Research is needed to compare our findings with a wider range of educators and investigate what feedback literacy means for resource design, PD, and classroom practice in secondary and undergraduate contexts.

CCS CONCEPTS

• Social and professional topics → K-12 education; **• Software and its engineering → Integrated and visual development environments**.

KEYWORDS

AI, ML, IDE, K-12 education, feedback literacy

ACM Reference Format:
Veronica Cucuiat and Jane Waite. 2024. Feedback Literacy: Holistic Analysis of Secondary Educators' Views of LLM Explanations of Program Error Messages. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653595

1 INTRODUCTION

The use of large language models (LLMs) to support learning to program has generated ideas and debate in the computer science education research community [30], in IT developer discussions [37], and emerging classroom practice advice [35]. This reveals a complex landscape for educators to navigate and heralds potential changes of what might be taught and how. Academic research has used LLM tools, for example, to generate code explanations for use in teaching activities [21], to assist beginner programmers' troubleshooting requests [13], and to create enhanced programming error messages [20]. Much of this work has focused on undergraduate settings, with limited research on school educators' views of the use of LLMs in teaching programming in secondary classrooms (US Grades 6–10). In 2023, we started to address this gap in a pilot study in England by asking two research questions: *RQ1: What are secondary educators' views on the potential classroom use of LLM program error message explanations? RQ2: In what ways can a feedback literacy perspective support the analysis of educators' views of potential classroom use of LLM program error message explanations?*

2 PRIOR WORK

A large language model (LLM) is "a complex mathematical representation of language that is based on very large amounts of data and allows computers to produce language that seems similar to what a human might say" [7]. How LLMs might be best used to support the teaching and learning of programming has yet to be agreed upon [3]. At an undergraduate level, studies examining how LLMs can generate code explanations to assist student code comprehension show significant potential [21, 34]. In one study with 1000 undergraduate students, LLM code explanations were rated as more accurate and easier to understand than student explanations [19]. Strengths and weaknesses have been highlighted in the output generated by LLMs in response to undergraduate beginner programmers' troubleshooting requests. For example, in one study, the LLM mostly generated sensible recommendations but was sometimes unreliable, and always included a solution even when explicitly prompted not to [13]. University-level educators' views, gathered from 20 programming instructors across nine countries on working with AI tools, range from banning them to integrating them to prepare students for future jobs [18]. For younger novice programmers, aged 10 to 17 years old, code-authoring performance has been cited to improve through using LLM tools [15]. Competencies students and educators need to use LLMs have been suggested, such as critically evaluating their output and potential biases [14]. Furthermore, questions remain on how LLMs can best support the needs of learners, the impact on students' learning and reliance on such tools [41].

A way of using LLMs is to enhance existing programming error messages (PEMs). PEMs represent diagnostic messages generated by compilers or interpreters when the code violates the specifications of the programming language [20]. They can help users progress towards a working program and support them in understanding the cause of the problem [22]. However, PEMs are often difficult to decipher [20] due to poor readability, including length, poor vocabulary, lack of sentence structure, and use of jargon [10].

Research efforts are ongoing to improve PEMs. In 2019, ten enhanced PEM guidelines were proposed from a synthesis of PEM research studies, including increased readability, reduced cognitive load, providing context for errors, using a positive tone, showing solutions or hints, and providing scaffolding for the user [1]. For example, increasing readability by avoiding jargon and using common vocabulary was one of the design principles applied in one study, which led to significantly shorter debugging times and higher self-reported scores of message usefulness for over 700 undergraduate students learning C [10]. Providing context for errors by linking the messages to programming concepts has been shown to help undergraduate students understand the cause of errors in a 13-week study, using a purpose-built programming tool [38]. In another undergraduate study, 100 students were presented with the original and enhanced error messages side-by-side, in a pedagogic Java editor, with no evidence that students were confused by seeing two versions of the same message [2]. Enhancing messages by using a positive tone has been recommended from a human-computer interaction perspective [39].

The inclusion of a possible solution in the enhanced message is debated [1]. Some suggest that distinct levels of help should be provided, from hints to solutions to accommodate different skills and needs [39]. The enhanced PEMs' role in scaffolding learners' knowledge has been disputed, due to the difficulty in identifying the underlying misconceptions which lead to students making errors [24]. However, efforts to improve the detection of misconceptions are ongoing, with one study developing SIDE-lib, a standalone, task-independent library to detect symptoms of misconceptions to support novice Python programmers into mainstream IDEs [11]. Overall, there is a lack of agreement on what makes a good PEM [22] and limited learning-theory-based research to investigate ways enhanced PEMs might be used for scaffolding learning [1].

Recently, researchers have evaluated LLM PEMs as being potentially effective, novice-friendly, more easy to interpret and action than the original PEMs [20]. PEMs play an important role in offering students immediate and consistent feedback [32], akin to formative assessment. Black and Wiliam[2010] argue in their seminal general education work on formative assessment that feedback provided by teachers has the most impact on student work if it includes guidance that students can engage with and practically use [4]. Scholars have been re-positioning the role of feedback in the context of higher education, from a transmission cognitive-based model which conceptualises feedback as information delivered to students [12], towards a learner-centred view which positions feedback as a process including the design of feedback and students' competencies to effectively engage with and utilise it [26].

McLean et al. [2015] in general education theory define four feedback types: a) *telling*, unidirectional 'correct' information to be packaged and transmitted from experts to passive students; b) *guiding*, as students begin to think about feedback as a two-way event in which the feedback acts as a pointer in the right direction; c) *developing understanding*, in which explanatory information is used by students for their own learning to understand what they are doing and why, beyond their immediate context; d) *opening up new perspectives*, making sense of information open to interpretation and critique to inform students' views [25].

Learner-centred feedback processes entail the development of student feedback literacy [8], defined as students' ability to comprehend, use and benefit from feedback processes [26], and teacher feedback literacy [9], in which effectiveness depends on what the learners bring, what the curriculum promotes, and what the environment affords [5]. Student feedback literacy encompasses students' capabilities and dispositions required to understand and use feedback in a way that is useful to them, encompassing four interrelated dimensions: *appreciating feedback processes, making judgements, taking action* and *managing affect* [8]. Teacher feedback literacy refers to teachers' abilities to design and implement feedback as a process that serves cognitive, pedagogical and social purposes [9]. Teacher feedback literacy concerns three dimensions: the *design* of learning materials to supports student feedback literacy; the *relational* support teachers provide students through emotional sensitivity, empathy and trust; and the *pragmatic* dimension that considers the inherent compromises involved in how teachers manage feedback practices [9].

3 METHOD

A free and publicly available interactive development environment (IDE) was augmented to support this study [31]. The augmented IDE was only available for the study, and it called the OpenAI GPT-3.5 LLM [28] to generate explanations of the IDE program error messages. Explanations were accessible to users via a question mark next to the standard error message and were displayed in a pop-up window. OpenAI GPT-3.5 was chosen as it was easy to set up, publicly available, and low-cost at the time of the study.

The prompt supplied to the LLM to generate the error message explanation was *"You are a teacher talking to a 12-year-old child. Explain the error {error} in the following Python code: {code}"*. The prompt was simple; our focus was to investigate the educators' views of an LLM output, not to evaluate prompt performance.

A teaching activity was selected for educators to follow as they used the augmented IDE. The activity was a publicly available, secondary-level Python programming task, that requires learners to adapt a program which prompts the input to a very simple "guess a lucky number" game [31]. Python was the chosen programming language, as it was the most used language for examinations at the time of the study [27]. Python is also popular in software engineering, making it likely that high volumes of Python code would have been used to train the OpenAI GPT-3.5 LLM [29].

Ethics approval was obtained in line with institutional and BERA guidelines [6]; related consent and data procedures were followed.

As this was a pilot study, expert educators were recruited to explore the feasibility of the study approach and to investigate more experienced educators' views. Eight educators active in computer science program education and known by the authors from previous collaborations were invited by email to take part. All eight participants were from England, four identified as male and four as female. The participants' length of time teaching computing at secondary level spanned from six to more than ten years.

Each educator was separately interviewed online, for between 30 minutes and one hour. Interviews were semi-structured and activity-based following a protocol of a) an introduction to the project; b) exploratory questions about the educator's thoughts on the use of LLM for learning to program; c) educator's experimentation of the IDE using the selected programming activity (and follow-up questions); d) questions on selected LLM-generated explanations (providing consistency across participants); e) general demographic questions; and f) final Q&A.

Interviews were transcribed using a professional service and uploaded to NVivo for analysis. Analysis followed a thematic qualitative approach [16]. High-level categories were deduced from the research questions and knowledge of the field [16, 23], then the text was coded, developing categories that were added, amended, or merged inductively [16, 23]. The first author coded all interviews and the second re-coded 25% of the data. The two authors resolved any differences, resulting in a consensus of themes and their definitions [16], with a Cohen's Kappa reliability score of 0.80 [17], which is considered high. The correlation of themes to feedback literacy theory was done jointly by both authors. Study resources and anonymised data are available on the study website [31].

4 RESULTS

Fifteen themes were derived from the data analysis and have been grouped into five reporting groups (Table 1). Overall, the educators' views of the LLM feedback were that, for the most part, a sensible explanation of the error messages was produced. All educators experienced at least one example of an invalid content, either when exploring the augmented IDE explanations or in the selected exemplar LLM responses (steps c) and d) of the interview protocol Section 3). Also, despite this not being prompted, a code solution was always included by the LLM in the explanation.

4.1 Content of explanations

All educators debated the implications of including a possible code solution in the explanation (theme 1). Mostly, they were concerned about students copying the code solution without understanding. On the other hand, some educators discussed the potential benefits students can derive from analysing a correct answer, with four educators suggesting a two- (or more) stepped approach in which the solution is only shown after the students have had one or more attempts at solving the error using only an explanation. All educators emphasised the importance of using key concept words consistently with students to support the development of discipline-related vocabulary (theme 2). Educators highlighted examples where they felt appropriate vocabulary was generated, as well as examples which lacked teaching-aligned terms:

> *"Also, it doesn't use the words or the expression 'data type', which is quite key for us when we're teaching. [..] I think there is some importance on it using the terminology that we use in education."* [Educator 2]

All educators were positive about the more detailed LLM explanation than the original error message (theme 3):

> *"But I do like that it doesn't just say what the correct answer is, it tries to explain why that's the right answer."* [Educator 4]

In addition, the educators positively commented on the clear presentation of the explanation, avoiding jargon, which they felt the students would more easily understand. Educators highlighted the different parts of the LLM output: a prediction of the students' goal; an explanation of why the current code was incorrect or incompatible with the goal; and why a certain fix might be applicable. However, educators identified inconsistencies with the parts of the explanation included and their sequence in different examples.

4.2 Format and style of explanations

Educators commented on how easy or difficult the explanations were to read for themselves or their students. For example, mentioning explanations being too verbose, spotting repetition, redundancy or too much information (theme 4). All but one educator said it was difficult to distinguish between code and text in the explanation, as both were displayed in the same font (theme 5):

> *"It mixes code and text in a way that is sometimes confusing. It's all in quotation marks. Do we know if the quotation marks are part of the code? Could it format them differently in a monospace font?"* [Educator 4]

Educators commented that the explanation tone was positive and encouraging (theme 6).

Five educators suggested that the feedback could be reformatted to highlight a before-after comparison (theme 7), to help students more clearly see the difference between the original code and that suggested in the explanation.

4.3 Validity of explanations

Educators' observations of the explanation's validity included that the LLM had 'made things up' by either introducing new syntax errors that didn't already exist (theme 8) or pointed out erroneous errors which did not appear in the code:

> *"And it's trying to fix something that doesn't need fixing."* [Educator 8]

Educators raised the detrimental impact of invalid explanations on students, such as confusing students or leading to misconceptions:

> *"They'll end up with misconceptions about the model, also about the code that they're trying to do, and just generally be confused and frustrated."* [Educator 8]

Educators highlighted explanations that were valid but that did not address the main learning objective related to the error, in that explanations were inconsistent in how accurately they identified the crux of the issue for students to resolve, pointing instead to other valid, but less relevant aspects of the code (theme 9).

4.4 How the learning process might be affected

Educators repeatedly emphasised the relationship between where students are on their learning journey versus student interest and motivation, as well as student ability to effectively use the feedback provided (theme 10). This included the knowledge required to understand and analyse the feedback, as well as the motivation required for students to use the messages effectively.

Table 1: Themes - Educators' practical considerations on using LLM explanations of program error messages

Reporting Groups	Themes	Cases (n=8)	Number of coded segments
Content of explanations	1. Possible code solution is always included	8	80
	2. Key concept words are generated inconsistently	8	43
	3. The explanation is detailed and avoids jargon	8	30
			153
Format and style	4. Lengthy and verbose explanation	8	46
	5. Program language elements are hard to distinguish from explanation	7	30
	6. Tone is positive and encouraging	7	30
	7. Student and explanation code solution should be displayed side-by-side	5	14
			120
Validity	8. Occasional invalid explanation could negatively affect students	8	80
	9. Explanation learning objectives are not always related to the error	7	28
			108
Learning process	10. Explanation effectiveness depends on student level and motivation	8	41
	11. Explanations are better than original PEMs but may cause dependency	8	38
	12. Students may fix more errors independently	5	12
			91
Teaching process	13. Educator PD needed on how LLMs work and implications for classroom use	8	35
	14. Opportunities for additional debugging teaching	7	18
	15. Student-educator interactions may be reduced	5	15
			68

Educators compared the program error message and the explanation, as well as the students' transition from one to the other (theme 11). Overall, the educators considered the explanation more helpful and meaningful than the program error message. However, the educators said it was important that the explanations should not replace the error message and that students needed support to be able to eventually transition to only using program error messages.

Five educators raised that some students might be able to fix their errors more independently using explanations than by using error messages (theme 12):

> *"I think what this enables me to do is to create that independent learning environment that gives a bit more information to those children that I would set independent learning tasks."* [Educator 5]

4.5 How the teaching process might be affected

All educators expressed the need to understand how explanations are generated, and as a consequence, what this might mean in how they should be used with students (theme 13):

> *"A summary flowchart of what data is sent, where it's sent to, how it's processed, how it's stored, what format it comes back in, and what the limitations and advantages are, would be really helpful."* [Educator 4]

Educators recognised opportunities to use the explanations to teach about debugging, for example using invalid explanations to discuss the criteria for good explanations and error messages (theme 14). Five educators highlighted that using explanations might reduce the time they spend helping students fix syntax errors (theme 15). On one hand, this can help save educators time:

> *"Oh, well, it definitely frees me up. It helps because it puts the onus back onto them to improve themselves and not just be reliant on me."* [Educator 7]

On the other hand, they expressed concern about reduced opportunities to interact with students and a negative effect on their understanding of student progress and reduced formative assessment. To mitigate against this, they emphasised the importance of being able to see student errors and explanations:

> *"If it's an online model, it's got to be a two-way street. The AI has to work for the teacher as well as the pupil. So, I need to know what are the error messages that they're getting, where are the pupils struggling."* [Educator 2]

We found that, for the most part, educators were excited about the potential of using LLMs to help explain error messages to learners, and most educators reported they would use them in their classrooms. However, perhaps as expected, using LLMs is unlikely to be a silver bullet in students' understanding of error messages, offering benefits and challenges in their current form.

5 DISCUSSION

5.1 An enhanced program error message lens

To answer RQ1: *What are secondary educators' views on the potential classroom use of LLM program error message explanations?*, we correlated the themes derived from the educators' commentary to the enhanced program error message (PEM) guidelines by Becker et al. [2019] and related literature. Six of the guidelines were discussed as practical considerations by the educators in our study (Table 2)

Firstly, the guideline of **increased readability** of PEMs [1] correlates with educators' comments on how the LLM explanations

Table 2: Correlation of PEM guidelines [1] to themes

Guideline [1]	Themes (Table 1)
Increased readability	3, 5
Reduce cognitive load	4, 7
Provide context to the error	8, 9, 11
Use a positive tone	6
Show solutions or hints	1
Provide scaffolding for user	2

were more readable, detailed and used plain English compared to standard PEMs (theme 3), but that readability was hindered as it was hard to distinguish between explanation text and program language elements (theme 5). The use of a common vocabulary and avoiding jargon is highlighted in the literature as an important aspect of improving PEM readability (e.g. [10]).

The guideline to **reduce cognitive load** can be correlated to educators' commentary on verbose explanations (theme 4) and the side-by-side comparison between student and solution code (theme 7). The educators commented that verbose feedback was, in part, caused by repetitive or redundant parts, which breaks the guideline of reducing cognitive load. To help students analyse the solution in a way which helps their understanding, educators proposed displaying students' code side-by-side with the solution code to highlight differences, which has been suggested to help reduce cognitive load [1] and found to not be confusing for students [2].

The **provide context to the error** guideline is reflected in the educators' discussions on instances in which the LLM explanations were either invalid (theme 8) or referred to unrelated learning objectives (theme 9), as well as when the educators debated the dangers of students' over-reliance on the LLM explanations (theme 11). The lack of accuracy and precision in PEMs is already heavily discussed in existing literature [39]. However, invalid LLMs can generate hallucinations which are entirely unrelated to the error, requiring further research into how students respond to invalid feedback. Building knowledge of core programming concepts by relating the explanations to learning objectives has been previously reported as helpful towards understanding the cause of the error [38]. To avoid over-reliance on LLM feedback, the educators' preference was for the explanations to be shown alongside the original error messages, something that has been suggested by others (e.g., [2]). Educators mentioned that LLM explanations were positive and encouraging (theme 6), adhering to the guideline of **using a positive tone**.

The guideline to **show solutions or hints** is reflected in theme 1 on the inclusion of a code solution. However, this was highly debated by educators, mirrored in literature (e.g., [1]). This highlights nuances around how a solution is offered to students, with educators and researchers suggesting a stepped approach [39].

The guideline of PEMs **providing scaffolding** for learners [1] is captured in the educators' comments on the inconsistent use of key concept words (theme 2). Whether LLMs can be useful to support existing efforts to identify students' misconceptions [11], or to enhance feedback using learning theories like semantic waves to scaffold learners' knowledge [40], remain areas of further research.

Table 3: Correlation of feedback theories to themes

Feedback theory	Themes (Table 1)
Feedback types [25]	
Telling	1, 5, 7
Guiding	1, 3, 4, 5, 7
Developing understanding	2, 3, 4, 5, 8, 9
Opening up new perspectives	8, 9
Student feedback literacy[8]	
Appreciating feedback process	1, 10
Making judgements	1, 2, 3, 5, 7, 8, 9, 10,11
Taking action	5, 7, 8, 9, 10, 11
Managing affect	6, 8, 10, 11, 12
Teacher feedback literacy[9]	
Design dimension	13,14
Relational dimension	15
Pragmatic dimension	15

What is significant as we look across the PEM guidelines/theme correlations is that not all themes have been allocated. Educators raised comments about the dependency of explanation effectiveness on the wider learning and teaching process that we found more difficult to correlate to the PEM guidelines.

5.2 A feedback theory lens

To answer RQ2: *In what ways can a feedback literacy perspective support the analysis of educators' views of potential classroom use of LLM program error message explanations?*, we correlated the educators' themes to feedback types [25], student feedback literacy [8], and teacher feedback literacy theory [9] (Table 3).

5.2.1 Feedback types. The LLM explanations in our study activities were, in the majority, **telling**, providing students with the final code straight away (theme 1). The majority of educators in our study preferred that this should not be so, rather that the explanation should **guide** and **develop understanding**, and if a final solution was to be given, it should be after several attempts of fixing the error without it. On **guiding** feedback, educators highlighted that error explanations should be detailed and jargon-free (theme 3), distinguish between program language elements and explanation text (theme 5), and use concept keywords (theme 2) to move towards **developing understanding**, enabling students to "feedforward" [40] for when they encounter future similar errors.

To **develop student understanding**, educators commented that there were significant risks when invalid LLM feedback was provided (theme 8), or when the learning objective addressed by the explanation was unrelated to the error the students made (theme 9). However, educators also saw these risks as opportunities for students, as students might be helped to take an active role in interpreting whether the feedback was correct and to **open up new perspectives**, using their previous knowledge or seeking alternative sources to scrutinise the explanation.

5.2.2 Student feedback literacy. Concerning **student feedback literacy [8]**, educators commented on all four dimensions (Table 3, Section 2), with a focus on the content, validity and learning process reporting groups. Educators' prediction that feedback effectiveness depends on students' knowledge and attitudes (theme 10) correlates to all four aspects of student feedback literacy.

Concerning students' **appreciation of feedback**, including a solution in the feedback (theme 1) risks inhibiting students from taking responsibility for developing their own understanding and reducing learner agency [8]. Conversely, several educators mentioned that a solution can offer students actionable information on why their code is broken. Further investigation is required into the ways students might access a solution that helps their understanding without reducing their agency.

The educators highlighted the importance of helping students **make judgements** about the LLM feedback, directly relating to their observations around the use of key-concept vocabulary (theme 2), detailed explanations avoiding jargon (theme 3), and invalid explanations (theme 8) or explanations unrelated to the errors made (theme 9). An important part of feedback literacy is helping students develop the language and concepts required to effectively participate in the discipline [36]. We argue that inconsistent use of key-concept words can hinder this development. In addition, students need extended opportunities to self-evaluate to develop their evaluative judgment [8]. We suggest jargon-free, detailed explanations offer such an opportunity in a way that is accessible to students to understand.

On students **taking action** to fix their code, educators predicted that when the feedback was invalid (theme 8) or the learning objective was unrelated to the error, this could hinder students from making sense of the error and cause them to take unhelpful action, potentially leading to increased frustration, or decreased confidence in themselves or the feedback received (theme 9).

In relation to students' capability to **manage affect**, educators predicted that the encouraging tone (theme 6) may help students feel more positively towards the feedback and more likely to engage with it, aligning directly with general feedback recommendations [8]. Also, educators highlighted the increased levels of control students may feel as a result of accessing a jargon-free explanation presented in an encouraging tone, that they can action independently (theme 12), echoing findings that learner independence may encourage learner agency [33].

5.2.3 Teacher feedback literacy. All themes relating to **teacher feedback literacy [9]** are found under the teaching process reporting group (Table 3). On the **design dimension** of teacher feedback literacy, educators requested professional development to learn about LLMs and how to use them in class (theme 13) and mentioned there would be opportunities to design classroom activities to teach debugging using LLMs (theme 14).

A specific set of comments aligned with the **relational** and **pragmatic dimensions** of teacher feedback literacy was that student-educator interactions may be reduced (theme 15). Using LLM explanations could serve a time-saving function, should it materialise in practice. Contrarily, educators voiced concerns about the detrimental impact on their relationship-building with students. Fewer interactions could limit the opportunities educators have to provide formative assessment and connect with students, negatively impacting their confidence and professional development (PD).

5.3 Combined lens

We suggest that combining enhanced PEM guidelines (a programming-specific lens) with feedback literacy theory (a general education lens) to analyse educators' views provides a new and more holistic approach to investigating the purpose and pedagogy of using LLM explanations in the teaching of programming.

From our combined view, we recommend that LLM content should be: encouraging, detailed, jargon-free, use keywords consistently, be in line with learning objectives, and a solution code should not be included or delayed. IDE design should ensure that educators can see how their students use LLM explanations, and enable users to manage invalid or unrelated explanations. To optimise programming teaching using LLMs, professional development and student learning materials combining feedback literacy, PEMs and LLMs should be researched, co-created and delivered.

6 LIMITATIONS AND FUTURE WORK

Only eight purposively sampled expert secondary computer science educators from England took part in this pilot and they predicted what their students might think. These participants' views may be biased. Therefore, future studies would benefit from widening the pool of educators and working directly with students. The performance of the AI model was not evaluated for the task, and the model prompt was simple, without much experimentation. Investigation of LLM prompts and tuning approaches to generate good explanation structures, such as semantic waves [40], and create tailored content and format of feedback is needed.

7 CONCLUSION

In this pilot study, we explored the views of eight experienced secondary educators on the use of LLM explanations of Python program error messages (PEM) for classroom use. Findings correlated with PEM guidelines on readability, reduced cognitive load, providing context to the error, using a positive tone, showing solutions or hints and providing scaffolding for the user. However, the PEM guidelines did not cover all the teacher-reported requirements, whereas feedback literacy theories did. Feedback type analysis revealed that educators preferred the LLM explanations to fulfil a *guiding* and *developing understanding* role of feedback rather than *telling*. Student feedback literacy highlighted that students need help to *make judgements* and *take action* with LLM explanations. For teacher feedback literacy, educators raised the importance of designing feedback with teachers as well as student feedback literacy in mind, and the need for professional development.

Incorporating feedback literacy when designing and using LLM explanations helps develop the content of the explanations, as well as the skills required by students and teachers to make effective use of these explanations, and focuses on the overall interaction. A combined feedback literacy and PEM guideline approach will be important for researchers and teachers to consider as LLMs become more prevalent in teaching and learning both in computer science and beyond.

ACKNOWLEDGMENTS

Thank you to the Raspberry Pi Foundation digital team for the development of the augmented IDE.

REFERENCES

[1] Brett Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris Mcdonald, Peter-Michael Osera, Janice Pearce, and James Prather. 2019. Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE-WGR '19)*. ACM, New York, NY, USA, 177–210. https://doi.org/10.1145/3344429.3372508

[2] Brett Becker, Graham Glanville, Ricardo Iwashima, Claire Mcdonnell, Kyle Goslin, and Catherine Mooney. 2016. Effective compiler error message enhancement for novice programming students. *Computer Science Education* 26, 2-3 (09 2016), 148–175. https://doi.org/10.1080/08993408.2016.1225464

[3] 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* (Toronto ON, Canada) *(SIGCSE 2023)*. ACM, New York, NY, USA, 500–506. https://doi.org/10.1145/3545945.3569759

[4] Paul Black and Dylan Wiliam. 2010. Inside the Black Box Raising Standards Through Classroom Assessment. 80 (09 2010). https://doi.org/10.1177/003172171009200119

[5] David Boud and Elizabeth Molloy. 2013. Rethinking models of feedback for learning. *Assessment & Evaluation in higher education* 38, 6 (2013), 698–712.

[6] British Educational Research Association. 2018. *Ethical guidelines for educational research* (4 ed.). British Educational Research Association, London. https://www.bera.ac.uk/resources/all-publications/resources-for-researchers

[7] Cambridge University Press. [n. d.]. LLM. *Cambridge Dictionary*. Accessed 26-03-2024. https://dictionary.cambridge.org/dictionary/english/large-language-model

[8] David Carless and David Boud. 2018. The development of student feedback literacy: enabling uptake of feedback. *Assessment & Evaluation in Higher Education* 43, 8 (2018), 1315–1325. https://doi.org/10.1080/02602938.2018.1463354

[9] David Carless and Naomi Winstone. 2023. Teacher feedback literacy and its interplay with student feedback literacy. *Teaching in Higher Education* 28, 1 (2023), 150–163.

[10] Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. ACM, New York, NY, USA, 480–486. https://doi.org/10.1145/3341525.3387384

[11] Abigail Evans, Zihan Wang, Jieren Liu, and Mingming Zheng. 2023. SIDE-Lib: A Library for Detecting Symptoms of Python Programming Misconceptions. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) *(ITiCSE 2023)*. ACM, New York, NY, USA, 159–165. https://doi.org/10.1145/3587102.3588838

[12] John Hattie and Helen Timperley. 2007. The power of feedback. *Review of educational research* 77, 1 (2007), 81–112.

[13] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of LLMs to Beginner Programmers' Help Requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1* (Chicago, IL, USA) *(ICER '23)*. ACM, New York, NY, USA, 93–105. https://doi.org/10.1145/3568813.3600139

[14] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, and et al. 2023. ChatGPT for Good? On Opportunities and Challenges of LLMs for Education. *Learning and Individual Differences* 103, 102274 (Jan 2023). https://doi.org/10.35542/osf.io/5er8f

[15] Majeed Kazemitabaar, Justin Chow, Carl Ma, Barbara Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. ACM, New York, NY, USA, Article 455, 23 pages. https://doi.org/10.1145/3544548.3580919

[16] Udo Kuckartz. 2022. *Qualitative Text Analysis: A Guide to Methods, Practice & Using Software*. Sage Publications, London. https://doi.org/10.4135/9781446288719

[17] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33, 1 (1977), 159–174.

[18] Sam Lau and Philip 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. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. ACM, New York, NY, USA, 106–121. https://doi.org/10.1145/3568813.3600138

[19] 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 LLMs. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) *(ITiCSE 2023)*. ACM, New York, NY, USA, 124–130. https://doi.org/10.1145/3587102.3588785

[20] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. 2022. Using LLMs to Enhance Programming Error Messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. ACM, New York, NY, USA, 563–569. https://doi.org/10.1145/3545945.3569770

[21] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating Diverse Code Explanations Using the GPT-3 LLM. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2* (Lugano and Virtual Event, Switzerland) *(ICER '22)*. ACM, New York, NY, USA, 37–39. https://doi.org/10.1145/3501709.3544280

[22] Guillaume Marceau, Kathi Fisler, and Shriram Krishnamurthi. 2011. Measuring the Effectiveness of Error Messages Designed for Novice Programmers. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. ACM, New York, NY, USA, 499–504. https://doi.org/10.1145/1953163.1953308

[23] Philipp Mayring. 2000. Qualitative Content Analysis. In *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research,*. Vol. 1 (2). Online. Issue Art.20. https://doi.org/10.17169/fqs-1.2.1089 Accessed 14-11-2022.

[24] Davin McCall and Michael Kölling. 2019. A New Look at Novice Programmer Errors. *ACM Trans. Comput. Educ.* 19, 4, Article 38 (jul 2019), 30 pages. https://doi.org/10.1145/3335814

[25] Angela J McLean, Carol H Bond, and Helen D Nicholson. 2015. An anatomy of feedback: a phenomenographic investigation of undergraduate students' conceptions of feedback. *Studies in Higher Education* 40, 5 (2015), 921–932. https://doi.org/10.1080/03075079.2013.855718

[26] Elizabeth Molloy, David Boud, and Michael Henderson. 2020. Developing a learning-centred framework for feedback literacy. *Assessment & Evaluation in Higher Education* 45, 4 (2020), 527–540.

[27] OCR. 2023. GCSE Computer Science: What programming language should I use for GCSE. Online. https://support.ocr.org.uk/hc/en-gb/articles/10066252791698-GCSE-Computer-Science-What-programming-language-should-I-use-for-GCSE- Accessed 19-12-2023.

[28] OpenAI. 2023. GPT-3.5. https://platform.openai.com/docs/models/gpt-3-5

[29] OpenAI. 2023. How ChatGPT and Our Language Models Are Developed. https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-language-models-are-developed

[30] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 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* (Turku,Finland) *(ITiCSE-WGR '23)*. ACM, New York, NY, USA, 108–159. https://doi.org/10.1145/3623762.3633499

[31] Raspberry Pi Foundation. 2023. Using large language models to explain programming error messages research study website. http://rpf.io/llm-pem-pilot

[32] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.

[33] Tracii Ryan, Michael Henderson, Kris Ryan, and Gregor Kennedy. 2023. Identifying the components of effective learner-centred feedback information. *Teaching in Higher Education* 28, 7 (2023), 1565–1582.

[34] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using LLMs. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1* (Lugano and Virtual Event, Switzerland) *(ICER '22)*. ACM, New York, NY, USA, 27–43. https://doi.org/10.1145/3501385.3543957

[35] Cory Stig. 2023. Can Generative AI Teach You to Code? It's Complicated. https://www.codecademy.com/resources/blog/can-chatgpt-ai-teach-you-to-code/

[36] Paul Sutton. 2012. Conceptualizing feedback literacy: knowing, being, and acting. *Innovations in Education and Teaching International* 49 (02 2012), 31–40. https://doi.org/10.1080/14703297.2012.647781

[37] The Chartered Institute for IT. 2023. Will AI replace software engineers? https://www.bcs.org/articles-opinion-and-research/will-ai-replace-software-engineers/

[38] Warren Toomey. 2011. Quantifying the incidence of novice programmers' errors. *School of IT, Bond University* (2011). https://api.semanticscholar.org/CorpusID:14453637

[39] V. Javier Traver. 2010. On Compiler Error Messages: What They Say and What They Mean. *Adv. in Hum.-Comp. Int.* 2010, Article 3 (jan 2010), 26 pages. https://doi.org/10.1155/2010/602570

[40] Jane Waite, Eirini Kolaiti, Meurig Thomas, and Karl Maton. 2023. Constructing feedback for computer science MCQ wrong answers using semantic profiling. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. ACM, New York, NY, USA.

[41] C. Zastudil, M. Rogalska, C. Kapp, J. Vaughn, and S. MacNeil. 2023. Generative AI in Computing Education: Perspectives of Students and Instructors. In *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–9. https://doi.org/10.1109/FIE58773.2023.10343467