Vol.2026-HCI-220 No.24
Vol.2026-UBI-92 No.24
2026/11/27

プログラミング教育におけるLLMフィードバックと学習者の対応行動の経年分析

関口 祐豊[1]　中村 聡史[1]

概要：プログラミング教育において，LLM は学習者に即時にフィードバックを提供できる一方，誤った指摘を生成してしまい学習者を困らせる場合もある．そのため，LLM の指摘の正確さだけでなく，学習者が指摘をどのように受け止め，修正に取り入れるかという，LLM との向き合い方を捉えることが重要である．本研究では，LLM による評価が含まれる 2024〜2026 年度のプログラミング演習の提出記録 2,400 件を対象に，学習者の対応行動と経年変化を分析した．具体的には，3 名の評価者が対応前後のコードの正誤，LLM の指摘内容の正誤，および指摘に沿う修正の有無を判定し，対応行動と再提出コードの正誤との関係を分析した．その結果，後の年度ほど指摘に沿う修正が増え，不備のあるコードの改善が進む一方，正しいコードの改悪も増えていることが明らかになった．これらの分析を通じて，学習者による LLM フィードバックの利用のあり方，および学生と LLM の関係の変化を議論する．

1. はじめに

プログラミング教育では，大規模言語モデル（Large Language Model，以下 LLM）によるフィードバックが，学習者の課題への取り組みを支える手段として授業に導入されている [1–5]．我々も，Processing [6] などを用いた視覚的・動的なプログラミング課題を対象に，LLM による一次評価を行う採点システム PP-Checker [7] を開発し，プログラミング演習で運用してきた．こうした支援では，LLM が生成したフィードバックをもとに，学習者がコードを修正するかどうかを判断する．そのため，フィードバックがコードの改善につながるかどうかは，出力の正確さに加え，学習者がその内容をどのように受け止め，コードに反映するかにも左右される．

LLM には，事実や与えられた情報に基づかない内容をもっともらしく生成するハルシネーションの問題があり [8]，学習者のコードに存在しない不備を指摘する可能性もある．また，学習者がフィードバックを採用するかどうかは，内容の正確さだけでなくの LLM への信頼にも左右され，誤った助言が受け入れられる場合もある [9,10]．学習者がフィードバックに沿ってコードを修正する場合，正しい指摘によって不備が解消されることもあれば，誤った指摘に従うことで正しいコードが不備のあるコードに改悪してしまうことも考えられる．一方，学習者が指摘を採用しない場合，誤った指摘による不要な変更を避けられる場合もあるが，正しい指摘が見過ごされ不備を残してしまうことも考えられる．さらに，LLM が指摘を示さない場合にも，コードが正しいとは限らず，それを鵜呑みにして修正しないことにより，不備が残ってしまう可能性がある．このように，LLM による支援では，フィードバックを採用することや修正が増えることが，そのまま適切なコードに至ることを意味するわけではない．

このような LLM フィードバックに対する学習者の対応は，支援を複数年度にわたって運用する中で変化する可能性がある．生成 AI の利用が学習場面に広がる中，LLM の評価を修正の判断材料として用いる傾向が強まれば，正しい指摘に沿う修正が増える一方，誤った指摘に沿う修正も増える可能性がある．こうした対応行動の変化は，LLM によるフィードバックがコードの改善を促す可能性と，誤った修正を招くリスクの双方に関わると考えられる．

そこで本研究では，LLM フィードバックに対する学習者の対応行動を年度間で比較し，その対応とフィードバック前後のコードの正誤との関係を明らかにする．これにより，学習者の適切な判断を支える LLM の教育的介入方法の設計に資する知見を得ることを目的とする．この目的に基づき，以下のリサーチクエスチョン（RQ）を設定する．

RQ：LLM フィードバックへの学習者の対応行動は年度間でどのように異なり，その対応はフィードバック前後のコードの正誤とどのように関係しているか．

この RQ を検討するため，PP-Checker を運用した 2024〜

[1] 明治大学
Meiji University

© 2026 Information Processing Society of Japan

2026 年度の明治大学総合数理学部先端メディアサイエンス学科 1 年次のプログラミング演習 I の提出記録から選定した 2,400 件を対象に，フィードバックに対する修正・却下行動を年度間で比較する．また，指摘の正誤と対応前後のコードの状態を踏まえ，それぞれの対応がコードの改善や改悪，正しい状態の維持などに結び付いているかを分析する．これにより，学生が LLM のフィードバックをどのように扱ってきたかを，対応行動の変化と提出コードの状態の両面から捉える．

2. 関連研究

2.1 LLM によるプログラミング学習支援システム

プログラミング学習では，LLM を用いて学習者のコードや質問に応じた説明・ヒントを提供する支援が授業に導入されている [11, 12]．課題や編集中のコードを参照して応答を生成する支援 [4] や，解答コードの直接提示を抑える設計 [13, 14] が提案されている．例えば CodeAid では，解答コードを直接提示しない支援を 12 週間の授業で評価し，誤ったフィードバックやコードの意図の取り違えを確認するとともに，学生が応答を検証・調整できる設計の必要性を述べている [1]．

LLM が生成するフィードバックの品質についても，課題コードやエラーメッセージを対象とした評価が行われている [2, 15]．Phung ら [16] は，GPT-4 が生成した詳細説明の有無による GPT-3.5 の模擬学生の修正結果を比較し，提示内容を検証する手法を提案した．Gipy [17] では，自動テストに基づくフィードバックを基準として LLM が生成するフィードバックの正確さを評価するとともに，実環境での導入を通じて，既存の自動評価を補完する可能性が検討されている．

教員による支援の調整に着目した ClassAid では，学生と AI のやり取りを可視化し，教員が AI の応答方法や提供の有無を授業中に調整できる [18]．我々が実現してきた PP-Checker も，LLM による一次評価と TA・教員による最終評価を組み合わせたシステムである [7]．

本研究は，こうしたシステムの授業内運用を，フィードバックと学生の修正行動との対応から捉える．PP-Checker の運用記録を用い，指摘に沿う修正に加え，指摘がない場合やコードを変更しない場合の対応も含めて，LLM フィードバックがどのように利用されたかを明らかにする．

2.2 AI フィードバックへの対応行動

学生が LLM フィードバックをどのように評価し，利用するかについては，質問紙や対話記録，発話思考法を用いた調査が行われている [3, 12, 19]．Xiao ら [20] は，4 段階のヒントを提供する LLM Hint Factory を用い，専門家が高品質と評価したヒントでも，学生の誤解などによって適切な修正につながらない事例を報告している．

プログラミング学習以外の人と AI による協調判断に関する研究でも，AI の提案を採用する割合が増えても，最終的な判断の正確さが高まるとは限らないことが報告されている [21]．また，AI の提案を確認するよう促す仕組みや，その確認に必要な手間によって，誤った提案が採用される割合が変化することが示されている [9, 22]．Schemmer らは，AI のフィードバックを受ける前後の回答を比較し，フィードバックへの対応の適切さを表す尺度を提案している [10]．

これらの研究を踏まえ，本研究では，LLM フィードバックへの対応行動と，対応後のコードの正誤を区別して分析する．具体的には，元コードと再提出コードがそれぞれ課題要件を満たすかを判定するとともに，指摘がある場合には，その内容に沿った修正が行われたかを確認する．また，LLM から指摘がない場合やコードを変更しない場合も含めて 2024〜2026 年度の記録を比較し，フィードバックへの対応行動の年度差と，対応前後のコードの正誤との関係を明らかにする．その結果に基づき，学習者が修正の要否を適切に判断できるよう支援する LLM の教育的介入方法について検討する．

3. 調査方法

3.1 PP-Checker と対象授業

PP-Checker は，LLM による一次評価と TA・教員による最終評価を組み合わせる採点システムであり，Processing を用いた視覚的・動的なプログラミング課題の評価に用いられている [7]．システムは．教員が設定した課題要件と採点基準に基づき，LLM が学習者のコードを一次評価し，提出前にフィードバックを返す．このとき，仕様を満たさない指摘がある場合は不正解，指摘がない場合は正解と判定される．学習者は，提示されたフィードバックを踏まえてコードを修正し，再評価を受けるか，追加の評価を受けずに提出するかを選択できる．TA や教員は，その提出されたコードの評価を行う．

対象は，PP-Checker を運用した 2024〜2026 年度前期のプログラミング演習 I である．履修者数は 2024 年度が 123 名，2025 年度が 121 名，2026 年度が 137 名であった．本稿では，この 3 年間の講義時間内の提出が求められる課題の記録を用いて，フィードバックへの対応と前後のコードの状態を捉える．

3.2 分析対象データ

各学習者の課題ごとに, LLM によるフィードバックと, その前後に提出されたコードをログから対応付けた．フィードバック前のコードを「元コード」，フィードバック後に提出されたコードを「再提出コード」と呼び，フィードバックと両コードを合わせて 1 件の記録として扱う．再提出コードには，元コードから変更されていないものも含む．

© 2026 Information Processing Society of Japan

両コードが課題要件を満たすかは，3.4 節で述べる人手評価により判定する．なお，複数の指摘を含むフィードバックでは，学習者が一部の指摘に沿ってコードを変更する一方，別の指摘には対応しないなど，指摘ごとに対応が異なる可能性がある．そこで本研究では，指摘とコード変更との対応関係を明確にするため，指摘が 1 件の記録と指摘のない記録のみを分析対象とし，指摘が 2 件以上の記録は分析対象から除外した．

対象データの収集期間は，2024 年度が 4 月 22 日から 7 月 15 日，2025 年度が 4 月 21 日から 7 月 7 日，2026 年度が 4 月 20 日から 7 月 13 日である．各年度から指摘が 1 件ある提出記録と指摘のない提出記録を 400 件ずつ合計 800 件を選定した（計 2,400 件）．課題ごとの選定数は，両者のうち候補が少ない方の件数を上限とし，最大 400 件になるまで課題ごとにランダムに割り当てた．

3.3 仮説

生成 AI の利用が学習場面に広がると，LLM の出力を参照しながら課題に取り組む機会も増えると考えられる．このような利用が一般化すれば，コードを変更するかどうかを判断する際にも，LLM の評価結果を手掛かりとする対応が増える可能性がある．しかし，評価を参照する機会が増えることと，その正確さを見極められることは同じではない．そのため，誤った評価も判断の手掛かりになってしまう可能性も考えられる．

そこで本研究では，年度の経過に伴い，LLM が不正解と判断した後に指摘に沿って修正する割合と，正解と判断した後にコードを維持する割合が，元コードの正誤にかかわらず高まると予測する．この予測を，元コードの正誤と LLM の判定による 4 つの条件に分け，以下の H1〜H4 として設定する．

H1：元コードが不正解であり，LLM も不正解と判定した場合，学習者が指摘に沿ってコードを修正する割合は，年度の経過とともに増加する．

H2：元コードが正解であるにもかかわらず，LLM が不正解と判定した場合，学習者が指摘に沿ってコードを修正する割合は，年度の経過とともに増加する．

H3：元コードが不正解であるにもかかわらず，LLM が正解と判定した場合，学習者がコードを修正せずに維持する割合は，年度の経過とともに増加する．

H4：元コードが正解であり，LLM も正解と判定した場合，学習者がコードを修正せずに維持する割合は，年度の経過とともに増加する．

3.4 コードおよびフィードバックの評価

コードとフィードバックの評価は，Processing の利用経験を 2 年以上有する 3 名の評価者が行った．評価者は，元コードと再提出コードがそれぞれ課題要件を満たすかを判定する．LLM からのフィードバックがある記録では，これに加えて，指摘内容の正誤と，指摘に沿う修正の有無を判定する（表 1）．元コードと再提出コードの正誤は，当該年度の課題資料に示された要件に基づいて判定する．LLM フィードバックについては，問題点の指摘が元コードと課題要件に照らして正しいかを判定する．指摘に沿う修正の有無は，本文と前後コードの差分を照合して判定する．指摘に対応する変更が行われていれば「修正あり」，変更がない場合や指摘と無関係な変更のみの場合は「修正なし」とする．



図 1 元コードと LLM フィードバック本文の正誤判定画面．



図 2 LLM の指摘に沿う修正の判定画面．

この評価には，課題資料とコード，フィードバック本文を提示する専用の Web システムを用いた．元コードの判定画面には課題資料と元コードを示し，指摘がある記録では本文も併記して，元コードと本文の正誤を同一画面で判定した（図 1）．指摘に沿う修正の判定には，本文と前後コードの差分を併せて提示した（図 2）．再提出コードの正誤を判定する画面には，課題資料と再提出コードを提示し，元コードやフィードバック本文，差分は表示しない設計とした．コードの動作確認を補助するため，Processing.js [23] による実行結果も参考情報として表示した．なお，Java と JavaScript の互換性には制約があるため，ブラウザで確認できない動作は，必要に応じてローカルの Processing で確認するように評価者に指示した．また，100 件および 500 件の判定時を，筆頭著者が評価者間に異なる判定傾向がないか確認した．

コードおよびフィードバックの評価における記録の提示順は，課題の切替負担を抑えるため，同一年度・同一課題の記録をまとめたブロックで構成する．ブロック内では，指摘あり・なしの記録をランダムに提示する．評価者の判定は，項目ごとに 2 名以上が選んだ回答を分析用の最終ラベルとした．

© 2026 Information Processing Society of Japan

表 1　評価項目と判定方法

項目	判定内容	取得方法
元コードの正誤	正解/不正解	元コードと課題資料を照合して判定する.
再提出コードの正誤	正解/不正解	再提出コードと課題資料を照合して判定する.
LLM の判定	正解/不正解	講義時の LLM 出力の結果から抽出する.
LLM フィードバックの正誤	正しい/誤り	LLM フィードバックを元コードと課題資料に照らして判定する.
指摘に沿う修正の有無	修正あり/修正なし	指摘と前後コード・差分を照合して判定する.

3.5　分析方法

H1〜H4 を検証するため，元コードの正誤，LLM による判定，フィードバック後の対応行動に基づいて提出記録を 8 分類する．対応行動の 8 分類と各仮説の関係を表 2 に示す．また，各判定項目の内容と取得方法を表 1 に示す．

元コードと再提出コードの正誤は，当該年度の課題要件を満たす場合に正解，満たさない場合に不正解とする．LLM による判定は運用時の記録から取得し，その正誤は元コードの正誤と照合して判断する．フィードバック後の対応行動については，LLM が不正解と判定した場合には指摘に沿う修正の有無を，正解と判定した場合にはコード変更の有無を確認する．なお，指摘がある記録では，フィードバック本文の正誤も別途判定する．

H1・H4 は元コードと LLM の判定が一致した場合の対応の変化を，H2・H3 は両者が一致しなかった場合の変化を検証する．ただし，LLM の不正解判定に沿う修正や，正解判定後のコード維持が増加しても，それが必ずしも適切な対応の増加を意味するわけではない．そこで，対応行動に加えて再提出コードの正誤も集計し，コードの改善や改悪，正しい状態の維持などとの関係を調べる．なお，Schemmer ら [10] の意思決定課題における 2 つの観点を参考に，LLM の正解・不正解の両判断を対象として，コードの改善と正解維持を評価する．

まず，元コードの正誤と LLM の判定による 4 つの条件ごとに，指摘に沿う修正率（H1・H2）とコード維持率（H3・H4）を年度別に求める．分母は，各年度の同じ条件に属する記録数とする．また，元コードが不正解の条件では，再提出時に正解となった割合を「改善率」，元コードが正解の条件では，再提出時にも正解だった割合を「正解維持率」とする．さらに，8 分類の各分類内でも再提出コードの正誤を集計し，対応行動とその結果を対応付ける．

経年変化は，3 年間で後の年度ほど対象の対応が生じやすくなるかで評価する．各条件について，指摘に沿う修正（H1・H2）またはコード維持（H3・H4）の有無を目的変数，年度（2024 年度=0，2025 年度=1，2026 年度=2）を説明変数とする二項ロジスティックモデルを用いる．同じ学生の複数記録の関連を考慮するため，一般化推定方程式（GEE）により，作業相関を独立とし，学生単位のロバスト分散で推定する．結果は 1 年あたりのオッズ比（OR）と 95%信頼区間で示す．OR が 1 より大きい場合，後の年度ほどその対応が生じやすい傾向を表す．増加方向の片側 Wald 検定を行い，H1〜H4 の p 値を Holm 法で調整し，OR が 1 を上回り，$p < 0.05$ の場合に仮説を支持する．

評価者間の一致度は，多数決による集約前の判定について，全員一致率と Fleiss の κ で示す [24].

4.　結果

4.1　分析対象と判定の一致度

分析対象の 2,400 件について，3 名の平均評価時間は 29.1 時間であった．表 3 に，コードおよびフィードバック評価結果の一致度を示す．全員一致率は 0.760〜0.824%，Fleiss の κ は 0.276〜0.666 であった．判定項目や指標による違いはあるものの，いずれの項目でも約 8 割の記録で 3 名の判定が一致しており，全体として一定の一致が得られた．

4.2　フィードバックへの対応と再提出結果

表 4 に，年度別の修正・維持の割合と H1〜H4 の検証結果を示す．図 3 に，H1〜H4 の各条件における年度別の対応行動と，再提出結果を示す．以下，年度別の割合は 2024，2025，2026 年度の順に示す．

元コードが不正解で LLM も不正解と判断した条件（H1）では，指摘に沿う修正率は 0.642, 0.728, 0.864 であった（図 3(a)）．年度間で有意な増加傾向が認められ，1 年あたりの OR は 1.85 であり（95%信頼区間：1.51〜2.27, $p < 0.001$），H1 は支持された．2024 年度から 2026 年度にかけて，修正率は 0.223 上昇した．この条件全体の改善率も 0.531，0.651，0.771 と増加し，年度差は +0.240 であった．指摘に沿って修正した記録に限ると，再提出コードの正解割合は 0.828，0.893，0.892 であった．H1 の対象コードにおいて修正があった場合においても 537 件中 67 件は不正解のままであり，修正がない 179 件は全て不正解と判定された．このことから，指摘に沿う修正はコードの改善と結び付く場合が多いものの，必ずしも課題要件の充足につながるわけではないことが確認された．

元コードが正解であるにもかかわらず LLM が不正解と判断した条件（H2）では，指摘に沿う修正率は 0.190，0.214，0.387 であった（図 3(b)）．1 年あたりの OR は 1.67 であり，有意な増加傾向が認められた（95%信頼区間：1.26〜

© 2026 Information Processing Society of Japan

表 2 学習者の対応行動の 8 分類.

分類	元コード	LLM による判定	LLM の指摘に沿った修正	対応する仮説
1	不正解	不正解	あり	H1
2	不正解	不正解	なし	
3	正解	不正解（誤った判断）	あり	H2
4	正解	不正解（誤った判断）	なし	
5	不正解	正解（誤った判断）	あり	H3
6	不正解	正解（誤った判断）	なし	
7	正解	正解（正しい判断）	あり	H4
8	正解	正解（正しい判断）	なし	

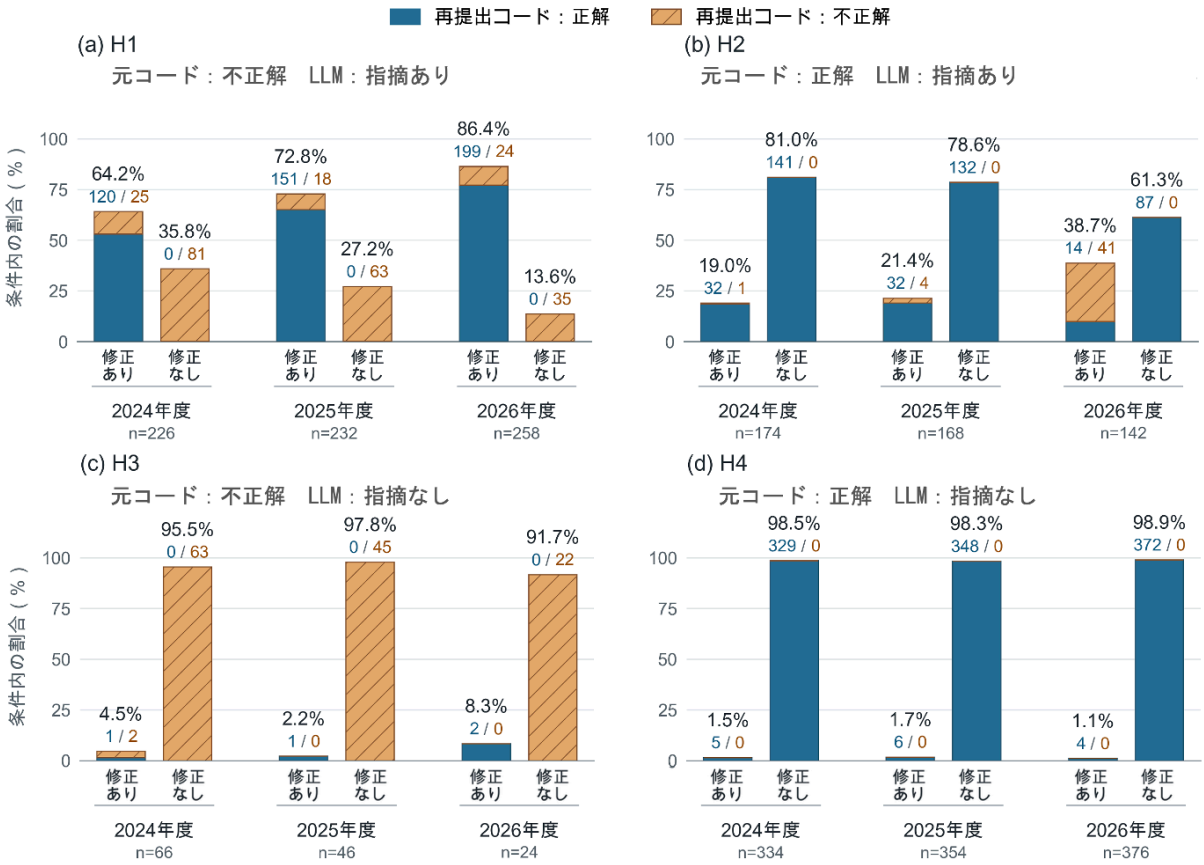

図 3 対応行動と再提出結果の年度比較．(a) ～ (d) は H1～H4 に対応する．色分けは再提出コードの正誤の内訳であり，棒上には対応の割合と正解件数／不正解件数を示す．

表 3 3 名の初回独立判定の一致度（3 年度）

判定項目	比較数	全員一致率	Fleiss の κ [24]
元コードの正誤	2,400	0.760	0.657
再提出コードの正誤	724	0.779	0.433
フィードバックの正誤	1,200	0.766	0.666
指摘に沿う修正	703	0.824	0.276

2.21，$p < 0.001$）ことから，H2 は支持された．2024 年度から 2026 年度にかけて，修正率は 0.198 上昇した．この条件全体の正解維持率は 0.994，0.976，0.711 であり，2026 年度は 2024 年度より 0.283 減少した．コードを修正した記録に限ると，再提出コードの正解割合は 0.970，0.889，0.255 であった．このように，元コードが正解であるにもかかわらず LLM の指摘に沿う修正が増加し，特に 2026 年度には修正後のコードが不正解となる割合が高かった．

元コードが不正解であるにもかかわらず LLM が正解と判断した条件（H3）では，コード維持率は 0.955，0.978，0.917 であった（図 3(c)）．1 年あたりの OR は 0.78 であり，有意な増加傾向は認められず，H3 は支持されなかった（95%信頼区間：0.23～2.58，$p = 0.660$）．3 年度を通じて，コードを変更しなかった 130 件はすべて再提出時にも不正解であった．コード変更があった記録も含めた条件全体の改善率は 0.015，0.022，0.083 であり，2026 年度の改善は 24 件中 2 件であった．しかし，不正解であるコードを LLM が正解と判定する件数は減少傾向にあることがわ

© 2026 Information Processing Society of Japan

表 4　対応行動の 3 年度分の比較.

条件（元コード／指摘）	対応	2024 年度	2025 年度	2026 年度	1 年あたりの OR［95%信頼区間］	p
H1（不正解／あり）	指摘に沿う修正	145/226（64.2%）	169/232（72.8%）	223/258（86.4%）	1.85［1.51, 2.27］	< 0.001
H2（正解／あり）	指摘に沿う修正	33/174（19.0%）	36/168（21.4%）	55/142（38.7%）	1.67［1.26, 2.21］	< 0.001
H3（不正解／なし）	コード維持	63/66（95.5%）	45/46（97.8%）	22/24（91.7%）	0.78［0.23, 2.58］	0.660
H4（正解／なし）	コード維持	329/334（98.5%）	348/354（98.3%）	372/376（98.9%）	1.18［0.66, 2.08］	0.578

かった.

元コードが正解で LLM も正解と判断した条件（H4）では, コード維持率は 0.985, 0.983, 0.989 であった（図 3(d)）. 1 年あたりの OR は 1.18 であり，有意な増加傾向は認められず，H4 は支持されなかった（95%信頼区間：0.66〜2.08, $p = 0.578$）. 一方，正解維持率はどの年度においても 1.000 であった. H3・H4 はいずれもコード維持が大部分を占めたが，H3 では不備が残存しており，H4 では正しいコードを維持していた.

4.3　指摘を含むフィードバックの正誤と指摘に沿う修正

LLM による正解・不正解の判定と，具体的な指摘内容の正誤は必ずしも一致しない. そこで，指摘がある 1,200 件を対象に，指摘内容の正誤と，指摘に沿う修正の有無との関係を分析した. フィードバック内容が正しい場合の修正率は，0.679，0.750，0.830 であった. 誤っている場合の修正率は，0.114，0.125，0.395 であった. いずれの年度でも正しい指摘に沿う修正率の方が高かったが，誤った指摘に沿う修正率も 2026 年度に大きく上昇していた.

4.4　コード修正の事例分析

本節では，LLM の指摘に沿う修正によって改善した事例と，正しかったコードが不正解となった事例を示す.

図 4(a) は，2024 年度のダイスの勝敗判定課題における H1 に対応する事例である. この課題では，大きい目を出した側を勝ち，同じ目なら引き分けと表示することが求められていた. 元コードでは，最初の条件が `diceA>=diceB` となっていたため，両者の目が等しい場合にも「A さんの勝ち」が出力され，引き分けを判定する分岐には到達しなかった. LLM はこの問題を正しく指摘し，再提出コードでは，最初の条件が `diceA==diceB` に変更され，出力も「引き分け」に修正されていた（13・17 行目）. さらに，次の条件が `diceA<diceB` に変更され，最後の分岐の出力も「A さんの勝ち」に修正されていた（19・29 行目）. これにより，勝敗と引き分けを課題要件に沿って判定できるようになり，再提出コードは正解となった. 同様の改善は，クリック判定の閾値を半径に合わせた例や，繰り返しの開始・終了値を指定範囲に合わせた例など，他の課題でも確認された.

図 4(b) は，2025 年度の四角形の移動課題における H2 に対応する事例である. この課題では，白と黒の四角形をそれぞれ `y=50` と `y=150` に配置し，両方を画面の左端から右へ移動させることが求められていた. 元コードはこの配置を満たし，両方の横位置は同じ式で求めていた. しかし，LLM は「白色の縦長四角形と黒色の縦長四角形が同じ位置に表示されている」と誤って指摘した. 再提出コードでは，黒い四角形の横位置を求める式に 100 が加えられ，移動の開始位置が変わっていた（12 行目）. この変更は指摘に沿う修正と判定されたが，正解だった元コードを改悪させてしまう行動として確認された. 同様の改悪は，他の課題でも確認された. 例えば，50 ミリ秒ごとに 1 ピクセル伸びるプログレスバーの課題では，LLM の指摘を受けて `frameRate(20)` が `frameRate(50)` に変更され，時間間隔の要件を満たさなくなった事例があった. また，ウィンドウサイズに関する指摘に沿って変更した結果，描画と操作の対応が崩れた事例もみられた.

5.　考察

5.1　指摘への対応とコードの改善・改悪

H1 では，2024 年度から 2026 年度にかけて，LLM の指摘に沿う修正率が 0.642 から 0.864 へ上昇し，コードの改善率も 0.531 から 0.771 へ上昇した. 一方，元コードが正解であるにもかかわらず LLM が不正解と判定した H2 でも，指摘に沿う修正率は 0.190 から 0.387 へ上昇したが，正解維持率は 0.994 から 0.711 へ低下した. このように，LLM の不正解評価に対して修正する行動は両条件で増加したものの，H1 ではコードの改善，H2 では正しいコードの改悪を伴っていた.

特に H2 では，各年度の分析対象に含まれる記録数が 174 件，164 件，142 件と減少した一方，元コードが正解から不正解に変化した記録は 1 件，4 件，41 件であった. 条件内での割合も 0.060，0.024，0.289 と上昇しており，誤った不正解評価を受けた後に正しいコードが損なわれる割合が増

© 2026 Information Processing Society of Japan

(a) 不正解から正解

元コード（不正解）

```
 1  void setup()
 2  {
 3    size(400, 300);
 4  }
 5
 6
 7
 8  void mousePressed()
 9  {
10    int diceA=(int) random(1, 7);
11    int diceB=(int) random(1, 5);
12
13*    if (diceA>=diceB)
14    {
15      println("A さん："+diceA);
16      println("B さん："+diceB);
17*     println("A さんの勝ち");
18    }
19    else if (diceA<=diceB)
20    {
21      println("A さん："+diceA);
22      println("B さん："+diceB);
23      println("B さんの勝ち");
24    }
25     else
26    {
27      println("A さん："+diceA);
28      println("B さん："+diceB);
29*     println("引き分け");
30    }
31  }
32
33  void draw()
34  {
35  }
```

再提出コード（正解）

```
 1  void setup()
 2  {
 3    size(400, 300);
 4  }
 5
 6
 7
 8  void mousePressed()
 9  {
10    int diceA=(int) random(1, 7);
11    int diceB=(int) random(1, 5);
12
13*    if (diceA==diceB)
14    {
15      println("A さん："+diceA);
16      println("B さん："+diceB);
17*     println("引き分け");
18    }
19*    else if (diceA<diceB)
20    {
21      println("A さん："+diceA);
22      println("B さん："+diceB);
23      println("B さんの勝ち");
24    }
25     else
26    {
27      println("A さん："+diceA);
28      println("B さん："+diceB);
29*     println("A さんの勝ち");
30    }
31  }
32
33  void draw()
34  {
35  }
```

(b) 正解から不正解

元コード（正解）

```
 1  void setup() {
 2    size(600, 300);
 3    background(255);
 4    noStroke();
 5  }
 6
 7  void draw() {
 8    background(255);
 9    drawStripes();
10
11    // 左端から右端まで移動する
      // アニメーション
12    int xPos1 =
        frameCount % (width + 20);
13*   int xPos2 =
        frameCount % (width + 20);
14
15    fill(255);
16    rect(xPos1, 50, 20, 100);
17
18    fill(0);
19    rect(xPos2, 150, 20, 100);
20  }
21
22  void drawStripes() {
23    fill(0);
24    int x = 0;
25    while (x < 600) {
26      rect(x + 10, 0, 10, 300);
27      x += 20;
28    }
29  }
```

再提出コード（不正解）

```
 1  void setup() {
 2    size(600, 300);
 3    background(255);
 4    noStroke();
 5  }
 6
 7  void draw() {
 8    background(255);
 9    drawStripes();
10
11    int xPos1 =
        frameCount % (width + 20);
12*   int xPos2 =
        (frameCount + 100) %
        (width + 20);
13    fill(255);
14    rect(xPos1, 50, 20, 100);
15
16    fill(0);
17    rect(xPos2, 150, 20, 100);
18  }
19
20  void drawStripes() {
21    fill(0);
22    int x = 0;
23    while (x < 600) {
24      rect(x + 10, 0, 10, 300);
25      x += 20;
26    }
27  }
```

図 4　指摘に沿う修正後に改善・改悪した 2 事例のコード．∗は変更された行を示す．

加していた．各年度の分析対象 800 件における再提出コードの正解割合は 0.785，0.838，0.848 と上昇していたが，その一方で H2 の正解維持率は大きく低下していた．この結果は，分析対象全体の正解割合が改善していても，特定の条件では LLM フィードバックへの対応によって正しいコードが損なわれる事例が増加し得ることを示している．

H1 と H2 で共に修正率が上昇した理由として，LLM への信頼の高まりが関わっている可能性がある．後の年度の受講者集団ほど，LLM の評価を修正の必要性を判断する根拠として受け入れやすくなっていれば，不備のあるコードへの見直しと，すでに正しいコードへの変更の双方が広がり得る．この結果から，LLM への信頼が修正行動を促す可能性がある一方で，LLM を修正の手掛かりとして利用する行動の広がりと，修正が必要な場面で適切に用いることを分けて捉える必要性が考えられる．

5.2　LLM による教育的介入の評価に向けた示唆

本研究では，指摘に沿う修正の有無と，再提出コードの正誤は対応しなかった．指摘に沿う変更があっても別の不備が残ることや，指摘とは異なる箇所の変更によって要件を満たすことがあり得るため，対応の有無から結果を推定することは適切ではない．本研究は，このずれを実際の記録で具体化し，修正行動と成果物の状態を個別に判定して対応付ける分析の意義を示した．コードを維持する同じ行動に，正しい状態の保持と不備の残存が含まれることから，変更のない記録も支援の評価対象となる．この視点を加えることで，LLM による支援を，修正を始める場面と見直しを終える場面の双方から検討できる．

5.3　修正の必要性を判断するための教育的介入

LLM を用いたプログラミング学習支援では，解答コードの直接提示を抑え，学習者自身が解決方法を考える余地を残す設計が採用されている [1, 13]．本研究の改悪事例では，LLM が示した誤った問題の指摘に沿って変更が行われ，すでに満たしていた課題要件が損なわれていた（図 4(b)）．この結果は，学習者に「どう修正するか」を考えさせるだけでなく，「そもそも修正が必要か」を判断できるよう支援することの重要性を示唆している．

修正の必要性を判断するには，指摘された問題が実際に生じているか，また，それが課題要件に反するかを確認する必要がある．例えば，図 4(b) では，2 つの四角形の実際の描画と課題で指定された配置を照合することで，同じ横位置にあることが修正を要する問題なのかを検討できる．そのため，LLM の指摘に関連する課題要件と，指摘を確かめるための入力・操作，実行結果を対応付けて提示する支援が考えられる．変更後にも，変更前のコードを保持して同じ条件で動作を比較できるようにすれば，指摘への対応によって課題で求められた動作がどう変わったかを確認できる．こうした支援は，指摘を採用するかどうかを，課題要件と実際の動作との対応に基づいて検討する機会として位置付けられる．

こうした判断支援は，指摘がないまま提出へ進む場面にも必要となる．本研究の結果では，LLM が正解と評価した後も，不正解コードの大半が維持されていた．この結果は，指摘のある箇所への対応を促す支援に加え，LLM が指摘しなかった要件を確かめる機会にも目を向ける必要性を示している．つまり，LLM が正解と判定した場合にも，課題要件に基づいて確認すべき条件を取り上げ，その条件を学習者が自然な形で再度見直せる設計が考えられる．こ

© 2026 Information Processing Society of Japan

れにより，期待される動作と実際の動作との照合を促し，両者の食い違いを検討する機会を設けることで，LLM が見逃した不備への気付きを支援できる可能性がある．

経年的な変化を踏まえると，LLM・学習者・TA・教員の役割分担も，継続運用の中で見直すことも重要と考えられる．本研究では，指摘に沿う修正が増加しているものの，コードの改善と正しいコードの改悪の双方が生じており，対応の増加だけを適切な利用の定着と捉えることはできない．そのため，LLM にどこまで指摘や確認の促しを担わせ，どの判断を学習者に委ね，どの場面で TA や教員が判断を支えるかを検討する必要がある．プログラミング教育への LLM の介入は，学習者と LLM の関わり方の変化を捉えながら，LLM による働きかけと人による支援を調整し続けることも求められる．

6. おわりに

本研究では，2024〜2026 年度の提出記録 2,400 件を対象に，LLM フィードバックへの対応行動と，対応前後のコードの正誤を年度間で比較した．その結果，正しい指摘と誤った指摘の双方に沿う修正が増加し，コードの改善と正しいコードの改悪がみられた．また，LLM が不備を見逃した場合には，多くのコードが不正解のまま維持されていた．これらを踏まえ，学習者の修正・提出の判断を支える教育的介入と，LLM・学習者・TA・教員の役割分担を継続的に見直す必要性を議論した．

本稿では，単一大学を対象に，年度ごとに異なる受講者集団を比較したものである．そのため，受講者や課題構成の違いを考慮しておらず，本稿の結果だけでは一般化は難しい．また，指摘が 1 件以下の記録に分析を限定しており，複数の指摘を選択的に取り入れる対応は扱えていない．

今後は，LLM からの指摘が複数ある条件においても調査を行い，LLM と学生の行動の関係性をより詳しく調査する．また，学習者と LLM，TA と LLM の関係を明らかにし，学習者の判断を支える教育的介入と，LLM・学習者・TA・教員の役割分担の設計につなげる．

参考文献

[1] Kazemitabaar, M. et al.: CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs, *CHI'24*, No. 650, pp. 1–20 (2024).
[2] Jacobs, S. et al.: Evaluating the Application of Large Language Models to Generate Feedback in Programming Education, *EDUCON*, pp. 1–5 (2024).
[3] Zhang, Z. et al.: Students' Perceptions and Preferences of Generative Artificial Intelligence Feedback for Programming, *AAAI* (2024).
[4] Bassner, P. et al.: Iris: An AI-Driven Virtual Tutor for Computer Science Education, *ITiCSE 2024*, p. 394–400 (2024).
[5] Heickal, H. et al.: A Classroom Study of LLM-generated Feedback Intervention in Introductory Programming, *Proceedings of the Impactful and Responsible AI Systems for Education Workshop*, Vol. 339, pp. 12–21 (2026).
[6] Reas, C. et al.: Processing: A Learning Environment for Creating Interactive Web Graphics, *SIGGRAPH'03* (2003).
[7] 関口祐豊ほか：PP-Checker: LLM と協働する曖昧性を許容した自動採点システム，コンピュータ ソフトウェア，Vol. 43, No. 1, pp. 15–26 (2026).
[8] Huang, L. et al.: A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions, *ACM Trans. Inf. Syst.*, Vol. 43, No. 2 (2025).
[9] Buçinca, Z. et al.: To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-assisted Decision-making, *Proc. ACM Hum.-Comput. Interact.*, Vol. 5, No. CSCW1 (2021).
[10] Schemmer, M. et al.: Appropriate Reliance on AI Advice: Conceptualization and the Effect of Explanations, *IUI '23*, p. 410–422 (2023).
[11] Liu, R. et al.: Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education, *SIGCSE 2024*, p. 750–756 (2024).
[12] Lyu, W. et al.: Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study, *L@S '24*, p. 63–74 (2024).
[13] Liffiton, M. et al.: CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes, *Koli Calling '23* (2024).
[14] George, S. et al.: NotebookGPT – Facilitating and Monitoring Explicit Lightweight Student GPT Help Requests During Programming Exercises, *IUI'24 Companion*, pp. 62–65 (2024).
[15] Azaiz, I. et al.: Feedback-Generation for Programming Exercises With GPT-4, *ITiCSE 2024*, p. 31–37 (2024).
[16] Phung, T. et al.: Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation, *LAK '24*, p. 12–23 (2024).
[17] Gabbay, H. et al.: Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming, *L@S '24*, p. 177–187 (2024).
[18] Zhang, G. et al.: ClassAid: A Real-time Instructor-AI-Student Orchestration System for Classroom Programming Activities, *CHI '26* (2026).
[19] Yang, S. et al.: Debugging with an AI Tutor: Investigating Novice Help-seeking Behaviors and Perceived Learning, *ICER '24*, p. 84–94 (2024).
[20] Xiao, R. et al.: Exploring How Multiple Levels of GPT-Generated Programming Hints Support or Disappoint Novices, *CHI EA '24* (2024).
[21] Bansal, G. et al.: Does the Whole Exceed its Parts? The Effect of AI Explanations on Complementary Team Performance, *CHI '21* (2021).
[22] Vasconcelos, H. et al.: Explanations Can Reduce Overreliance on AI Systems During Decision-Making, *Proc. ACM Hum.-Comput. Interact.*, Vol. 7, No. CSCW1 (2023).
[23] Salga, A. et al.: Processing.js: Sketching with Canvas, *SIGGRAPH '11*, No. 15, p. 1 (2011).
[24] Fleiss, J. L.: Measuring Nominal Scale Agreement Among Many Raters, *Psychological Bulletin*, Vol. 76, No. 5, pp. 378–382 (1971).

© 2026 Information Processing Society of Japan