ĐỀ XUẤT HỆ THỐNG TRỢ GIẢNG LẬP TRÌNH DỰA TRÊN AI HỖ TRỢ PHÁT HIỆN LỖI, GIẢI THÍCH MÃ VÀ PHÁT TRIỂN NĂNG LỰC GIẢI QUYẾT VẤN ĐỀ

PHẠM THẾ ANH
VŨ ĐỨC HUY
Đại học Công nghiệp Hà Nội

Nhận bài ngày 23/9/2026. Sửa chữa xong 19/10/2026. Duyệt đăng 24/10/2026.

Abstract

This paper proposes an AI-based programming tutoring system that assists learners in detecting programming errors, understanding their causes, and iteratively improving their code. Instead of simply providing corrected code, the system employs a learning cycle comprising task definition, code and test-case submission, error analysis, evidence-based explanations, targeted hints, and learner verification. Its architecture integrates code-structure analysis, test execution, and a language model for feedback generation, together with a control layer designed to minimize inappropriate or unsupported AI suggestions. The paper presents the theoretical foundations, system architecture, operational workflow, feedback model, interface prototype, an illustrative Python programming scenario, and a proposed experimental design. As no classroom experiment has yet been conducted, the findings are limited to system design and illustration rather than quantitative evidence of learning effectiveness. The proposed architecture provides a foundation for future research on programming assessment, personalized feedback, and the responsible use of AI in information technology education.

Keywords: *AI tutoring, automated feedback, code analysis, error detection, problem-solving competence, programming education.*

1. Đặt vấn đề

Trong đào tạo công nghệ thông tin, lập trình là một lĩnh vực đòi hỏi người học đồng thời vận dụng kiến thức cú pháp, cấu trúc dữ liệu, thuật toán và kỹ năng phân tích lỗi. Một chương trình có thể không chạy do lỗi cú pháp, chạy nhưng cho kết quả sai, hoặc chỉ thất bại ở một số trường hợp biên. Vì vậy, phản hồi trong quá trình học không nên dừng ở thông báo "đúng/sai", mà cần giúp người học hiểu bằng chứng nào dẫn tới kết luận và cách tự kiểm tra giả thuyết sửa lỗi.

Sự phát triển của các mô hình ngôn ngữ tạo ra khả năng hỗ trợ viết, giải thích và sửa mã với tốc độ cao. Tuy nhiên, phản hồi sinh bởi AI có thể thiếu căn cứ, bỏ qua yêu cầu cụ thể của bài tập hoặc khiến người học sao chép lời giải thay vì tự phân tích. UNESCO nhấn mạnh cách tiếp cận lấy con người làm trung tâm, trong đó việc sử dụng AI trong giáo dục cần đi kèm các yêu cầu về an toàn, bảo vệ dữ liệu và thiết kế sư phạm phù hợp (Miao & Holmes, 2023). NIST cũng xem quản trị, lập bản đồ, đo lường và quản lý rủi ro là các chức năng cốt lõi khi phát triển hệ thống AI đáng tin cậy (Tabassi, 2023).

Trong thực tế giảng dạy, một giảng viên thường phụ trách các lớp thực hành đông người, nên sinh viên phải chờ lâu để được xem mã hoặc chỉ nhận những nhận xét ngắn như "kiểm tra lại vòng lặp". Khi phản hồi chậm, họ dễ chuyển sang tìm lời giải có sẵn hoặc nhờ công cụ AI viết lại toàn bộ chương trình. Từ vấn đề trên, bài báo đề xuất một hệ thống trợ giảng lập trình theo hướng "AI hỗ trợ – người học kiểm chứng". Mục tiêu không phải xây dựng một công cụ tự động làm bài thay người học, mà là tạo ra môi trường trong đó lỗi được biến thành đối tượng học tập. Bài báo tập trung vào bốn nội dung: (i) xác định yêu cầu và nguyên tắc thiết kế hệ thống; (ii) đề xuất kiến trúc kết hợp phân tích mã, kiểm thử và

Email: huyvd@fit-haui.edu.vn

AI; (iii) xây dựng quy trình phản hồi nhiều bước; và (iv) minh họa bằng một bài toán Python cùng các bảng tiêu chí đánh giá.

2. Cơ sở lý thuyết và các nghiên cứu liên quan

2.1. Phản hồi và hỗ trợ theo giàn giáo

Hattie và Timperley cho rằng phản hồi hiệu quả cần trả lời ba câu hỏi: người học hướng tới đâu, hiện đang ở đâu và bước tiếp theo là gì (Hattie & Timperley, 2007). Ba câu hỏi này được phản ánh trong cấu trúc "bằng chứng – diễn giải – hành động tiếp theo" của hệ thống đề xuất. Khái niệm vùng phát triển gần nhất của Vygotsky (Vygotsky, 1978) và khái niệm "giàn giáo" của Wood, Bruner và Ross (Wood et al., 1976) cho thấy hỗ trợ nên vừa đủ, tăng khi người học gặp khó và rút dần khi họ tiến bộ. Mô hình bốn mức phản hồi L1–L4 ở Mục 4.2 là cách vận hành hóa tinh thần đó. Từ góc độ nhận thức, Sweller chỉ ra rằng thử – sai không định hướng có thể làm quá tải bộ nhớ làm việc của người mới học (Sweller, 1988); do đó, phản hồi tốt nên giảm tải không cần thiết (chỉ đúng vùng mã cần xem) nhưng vẫn để người học tự thực hiện phần suy luận cốt lõi.

2.2. Hệ thống dạy kèm thông minh và phản hồi tự động cho bài lập trình

Các bộ dạy kèm nhận thức theo dõi từng bước giải của người học và đưa gợi ý đúng thời điểm (Anderson et al., 1995). VanLehn kết luận rằng dạy kèm chi tiết ở mức bước giải có hiệu quả gần với dạy kèm một–một của con người (VanLehn, 2011). Với bài tập lập trình, Keuning và cộng sự nhận thấy phần lớn công cụ hiện có dừng ở việc báo lỗi và kiểm thử đúng/sai, trong khi gợi ý theo ngữ cảnh và đánh giá hiệu quả sư phạm còn hạn chế (Keuning et al., 2019); Ala-Mutka cũng phân biệt đánh giá động (chạy test) và đánh giá tĩnh (phân tích cấu trúc mã) (Ala-Mutka, 2005). Hệ thống đề xuất kết hợp cả hai để tạo bằng chứng.

2.3. Mô hình ngôn ngữ lớn và AI giải thích được

Các mô hình ngôn ngữ lớn (Brown et al., 2020) có khả năng sinh mã, giải thích mã và sửa lỗi, nhưng cũng làm lung lay các hình thức đánh giá truyền thống (Prather et al., 2023). CodeAid là một ví dụ về trợ lý được thiết kế để giải thích và gợi ý thay vì trả mã hoàn chỉnh, song vẫn gặp khó khi phản hồi chưa sát yêu cầu bài tập (Kazemitabaar et al., 2024). Về giải thích, các phương pháp như LIME (Ribeiro et al., 2016) và SHAP (Lundberg & Lee, 2017) giải thích dự đoán của mô hình học máy, còn Barredo Arrieta và cộng sự nhấn mạnh rằng giải thích phải gắn với đối tượng và mục đích sử dụng (Barredo Arrieta et al., 2020; Molnar, 2022). Trong trợ giảng lập trình, điều cần giải thích là vì sao chương trình của người học sai; vì vậy nguồn giải thích đáng tin cậy nhất là bằng chứng từ chính chương trình như dòng mã, giá trị biến và test case thất bại.

2.4. Khoảng trống và định vị đề xuất

Các công cụ chấm bài tự động mạnh về kiểm thử nhưng yếu về phản hồi sư phạm; các trợ lý dựa trên mô hình ngôn ngữ mạnh về diễn đạt nhưng thiếu cơ chế ràng buộc bằng chứng và điều tiết mức hỗ trợ. Bảng 1 so sánh ba hướng tiếp cận với hệ thống đề xuất.

Bảng 1: So sánh các hướng tiếp cận hỗ trợ phản hồi cho bài tập lập trình

Tiêu chí	Chấm bài tự động	Trợ lý LLM thuần	Hệ thống đề xuất
Cơ sở bằng chứng	Kết quả test, log	Văn bản mã và lời nhắc	Test, AST, log, đề bài
Giải thích nguyên nhân	Hầu như không	Có, khó kiểm chứng	Có, gắn với bằng chứng
Điều tiết mức hỗ trợ	Không	Phụ thuộc lời nhắc	Bộ điều phối L1–L4
Nguy cơ đưa đáp án	Thấp	Cao	Được kiểm soát

3. Phương pháp nghiên cứu

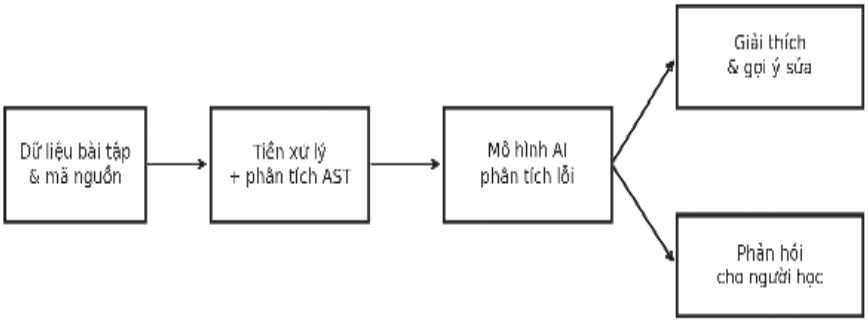
Bài báo sử dụng phương pháp nghiên cứu thiết kế ở mức kiến trúc và đặc tả chức năng. Cách tiếp cận này phù hợp với giai đoạn xây dựng mô hình khái niệm trước khi tiến hành thực nghiệm. Nội dung được tổng hợp từ các hướng nghiên cứu về AI đáng tin cậy, AI giải thích được, hỗ trợ lập trình và phản hồi trong môi trường học tập.

Quy trình thiết kế gồm năm bước: (1) phân tích yêu cầu của hoạt động học lập trình; (2) xác định

các nguồn bằng chứng mà hệ thống phải sử dụng trước khi sinh phản hồi; (3) thiết kế kiến trúc xử lý mã và kiểm thử; (4) thiết kế cơ chế phản hồi theo mức độ hỗ trợ; và (5) xây dựng kịch bản minh họa để kiểm tra tính nhất quán của luồng xử lý. Trong thiết kế này, mô hình ngôn ngữ không được coi là nguồn chân lý duy nhất. Phản hồi nên được đối chiếu với mã nguồn, kết quả thực thi và bộ kiểm thử trước khi hiển thị cho người học.

Khung XAI được sử dụng như cơ sở để phân biệt giữa "gợi ý có thể kiểm chứng" và "lời giải thích chỉ dựa trên ngôn ngữ". Các nghiên cứu tổng quan về XAI chỉ ra rằng tính giải thích cần được xem xét cùng bối cảnh, mục tiêu sử dụng và giới hạn của mô hình (Barredo Arrieta et al., 2020). Vì vậy, hệ thống đề xuất ưu tiên cung cấp bằng chứng cụ thể như dòng mã, test case thất bại, giá trị biến và điều kiện dẫn tới lỗi. Từ Mục 2, nghiên cứu rút ra bốn yêu cầu thiết kế: mọi nhận định phải gắn với bằng chứng kiểm chứng được; mức hỗ trợ tăng dần theo nhu cầu; các lựa chọn mà đề bài chưa quy định thuộc về người học; và mọi phản hồi cùng quyết định của hệ thống phải được ghi lại để truy vết.

Hình 1: Kiến trúc khái niệm của hệ thống trợ giảng lập trình dựa trên AI



Vòng lặp học tập: người học sửa mã → chạy lại → nhận phản hồi → tự đánh giá

4. Kết quả thiết kế hệ thống

4.1. Đặc điểm chức năng của hệ thống

Hệ thống được tổ chức thành bốn vùng chức năng chính: tiếp nhận bài nộp; phân tích mã và kiểm thử; sinh phản hồi; và theo dõi vòng lặp sửa mã. Cách tổ chức này cho phép tách nhiệm vụ tính toán khỏi nhiệm vụ giao tiếp với người học.

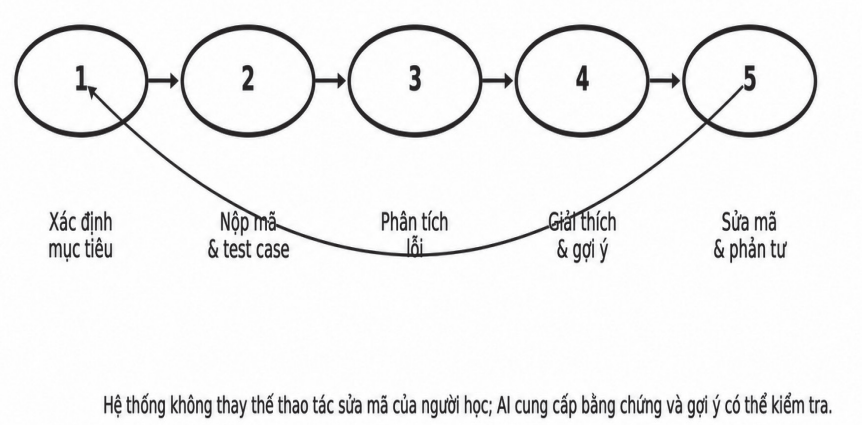
Thành phần tiếp nhận bài nộp lưu mã nguồn, ngôn ngữ lập trình, yêu cầu bài toán và bộ kiểm thử. Thành phần phân tích mã có thể sử dụng cây cú pháp trừu tượng (AST), thông tin lỗi khi biên dịch hoặc thực thi, cùng kết quả của các test case. Thành phần AI nhận các bằng chứng đã được chọn lọc thay vì chỉ nhận toàn bộ mã nguồn. Cuối cùng, bộ điều phối phản hồi quyết định mức gợi ý: chỉ nêu vị trí lỗi, đưa câu hỏi định hướng, gợi ý chiến lược hoặc giải thích sâu hơn. Việc chuyển "gói bằng chứng" thay vì toàn bộ mã cho mô hình AI giúp giảm nguy cơ mô hình suy diễn những lỗi không tồn tại, cho phép kiểm tra chéo từng nhận định và giảm lượng dữ liệu gửi đi.

4.2. Quy trình hỗ trợ học lập trình

Quy trình được thiết kế thành năm bước liên kết. Bước 1 xác định mục tiêu giúp hệ thống biết người học đang giải quyết loại nhiệm vụ nào. Bước 2 tiếp nhận mã và test case để tạo bằng chứng ban đầu. Bước 3 phân tích lỗi bằng cả thực thi và phân tích cấu trúc mã. Bước 4 chuyển các bằng chứng thành phản hồi có mức độ. Bước 5 yêu cầu người học sửa, chạy lại và giải thích thay đổi.

Điểm khác biệt quan trọng là hệ thống không tự động chuyển ngay từ lỗi sang mã hoàn chỉnh. Mỗi phản hồi nên chứa ba lớp: bằng chứng, diễn giải và hành động tiếp theo. Ví dụ, thay vì trả về toàn bộ hàm đã sửa, hệ thống có thể chỉ ra rằng danh sách rỗng làm mẫu số bằng 0, yêu cầu người học đề xuất điều kiện kiểm tra, sau đó cho phép người học chạy lại test case.

Hình 2: Quy trình hỗ trợ học lập trình theo vòng lặp phản hồi



Bảng 2: Mức độ phản hồi đề xuất

Mức	Mục tiêu	Ví dụ phản hồi	Khi sử dụng
L1 – Định vị	Giúp người học tìm vị trí cần xem	"Kiểm tra điều kiện ở dòng 8."	Người học mới phát hiện lỗi
L2 – Gợi ý	Định hướng cách suy nghĩ	"Điều gì xảy ra khi danh sách rỗng?"	Cần kích hoạt tự sửa
L3 – Giải thích	Nêu quan hệ nguyên nhân–kết quả	"len(a)=0 nên phép chia không xác định."	Người học cần hiểu lỗi
L4 – Mẫu sửa	Minh họa cách sửa sau khi đã thử	Đưa một phương án sửa và giải thích	Chỉ dùng khi L1–L3 chưa đủ

Mức phản hồi được chọn dựa trên tín hiệu từ kho tiến trình: số lần nộp liên tiếp không làm giảm số test thất bại, việc người học đã trả lời câu hỏi phản tư hay chưa, và mức hỗ trợ đã nhận ở lần trước. Quy tắc cơ bản là bắt đầu ở L1 hoặc L2; chuyển lên mức cao hơn khi người học đã nộp lại ít nhất hai lần mà không tiến triển; hạ xuống khi họ tự sửa thành công. Mức L4 chỉ kích hoạt khi người học chủ động yêu cầu và đã trải qua một chu kỳ L2 hoặc L3 cho cùng lỗi, đồng thời việc xem mẫu được ghi lại. Các ngưỡng này là tham số cấu hình theo học phần và cần được hiệu chỉnh bằng dữ liệu thực nghiệm; khi đã có đủ dữ liệu, bài toán chọn mức hỗ trợ có thể được xem xét bằng các phương pháp học tăng cường (Sutton & Barto, 2018).

4.3. Mô hình dữ liệu phản hồi

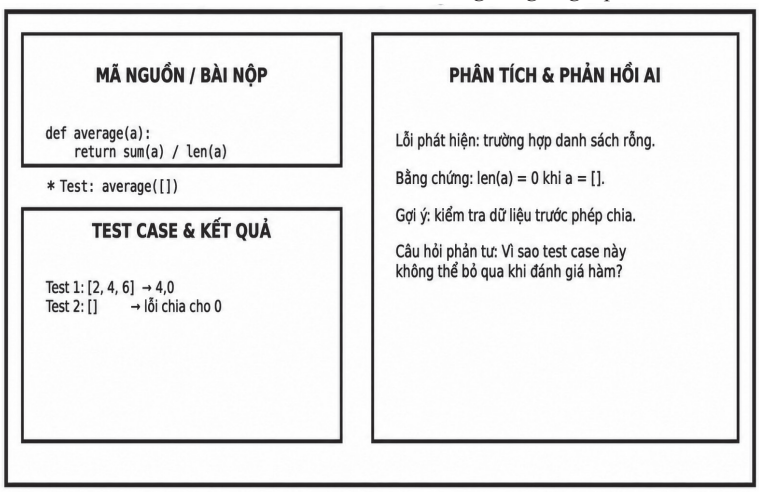
Để hạn chế phản hồi chung chung, mỗi thông báo được biểu diễn theo cấu trúc gồm: mã lỗi, bằng chứng, nguyên nhân, mức độ chắc chắn, gợi ý hành động và câu hỏi phản tư. Cấu trúc này cho phép giáo viên xem lại quá trình và phân biệt lỗi của người học với lỗi do công cụ phân tích.

Một phản hồi mẫu có thể được mô tả như sau: ErrorType = RuntimeError; Evidence = "test case [] thất bại"; Cause = "mẫu số bằng 0"; Hint = "xem xét điều kiện khi độ dài danh sách bằng 0"; Reflection = "hãy xác định đầu vào tối thiểu mà hàm phải xử lý". Với thiết kế này, AI được yêu cầu diễn đạt dựa trên bằng chứng đã có, thay vì tự tạo ra dữ kiện. Khi bằng chứng đến từ test thất bại và dấu vết ngăn xếp, mức chắc chắn cao; khi nhận định chỉ dựa vào cấu trúc mã, mức chắc chắn thấp và phản hồi phải được diễn đạt dưới dạng câu hỏi chứ không phải khẳng định.

4.4. Giao diện minh họa

Giao diện đề xuất chia màn hình thành ba vùng: mã nguồn, kết quả kiểm thử và phân tích AI. Việc đặt ba vùng cạnh nhau giúp người học đối chiếu trực tiếp giữa chương trình, bằng chứng và lời giải thích. Giao diện cũng cần hiển thị lịch sử các lần sửa để người học nhận biết mình đã thay đổi điều gì và thay đổi đó có làm giảm lỗi hay không. Nút "xin gợi ý thêm" nên chỉ khả dụng sau khi người học nhập giả thuyết của mình, nhằm khuyến khích suy nghĩ trước khi nhận hỗ trợ.

Hình 3: Giao diện minh họa cho hệ thống trợ giảng lập trình



4.5. Bộ tiêu chí đánh giá sản phẩm

Ở giai đoạn nghiên cứu tiếp theo, hệ thống cần được đánh giá theo năm nhóm tiêu chí. Thứ nhất, độ chính xác kỹ thuật được xác định bằng tỉ lệ lỗi được định vị đúng khi so sánh với bộ lỗi chuẩn, qua đó đánh giá năng lực phát hiện lỗi của hệ thống. Thứ hai, chất lượng phản hồi được hai chuyên gia đánh giá theo rubric về tính đúng, rõ ràng và phù hợp với trình độ người học nhằm xác định giá trị sư phạm. Thứ ba, tính giải thích được kiểm tra trên từng phản hồi dựa vào việc có nêu bằng chứng và nguyên nhân hay không, từ đó xác định khả năng hạn chế những nhận định thiếu căn cứ. Thứ tư, hành vi học tập được đo bằng số lần người học tự sửa trước khi xem mẫu, thu thập từ nhật ký hệ thống nhằm đánh giá mức độ chủ động. Cuối cùng, mức độ an toàn được đánh giá thông qua kiểm thử bảo mật, tập trung vào khả năng ngăn mã nguy hiểm thực thi ngoài sandbox và giảm thiểu rủi ro đối với hệ thống.

4.6. Phân loại lỗi và chiến lược phản hồi

Để bộ điều phối quyết định nhất quán, hệ thống cần một bảng phân loại lỗi đủ gọn để triển khai. Hệ thống phân loại đề xuất năm nhóm lỗi thường gặp ở học phần nhập môn, kèm nguồn bằng chứng và chiến lược mặc định; giáo viên bộ môn có thể hiệu chỉnh theo ngôn ngữ và nội dung học phần. Ranh giới giữa các nhóm không tuyệt đối (lỗi chia cho 0 có thể do bỏ sót trường hợp biên hoặc hiểu sai yêu cầu), nên nhóm lỗi được gán kèm mức chắc chắn và có thể được giáo viên chỉnh sửa.

Bảng 3: Phân loại lỗi và chiến lược phản hồi mặc định

Nhóm lỗi	Biểu hiện	Nguồn bằng chứng	Chiến lược mặc định
Cú pháp	Không phân tích được (thiếu dấu, thụt lề sai)	Thông báo phân tích cú pháp	L1: nêu vị trí
Thực thi	Ngoại lệ khi chạy (chia cho 0, chỉ số ngoài phạm vi)	Dấu vết ngăn xếp, test	L2: hỏi điều kiện gây ngoại lệ
Trường hợp biên	Đúng với dữ liệu thường, sai với rỗng hoặc cực trị	Test case biên	L2: liệt kê đầu vào tối thiểu
Logic thuật toán	Kết quả sai ở nhiều test	So sánh đầu ra, theo dõi biến	L2→L3: theo dõi từng bước
Hiểu sai yêu cầu	Chạy ổn định nhưng giải bài toán khác	Đối chiếu đề bài	L2: diễn đạt lại đề

4.7. Ràng buộc đối với mô hình ngôn ngữ

Mô hình ngôn ngữ chỉ được dùng ở tầng diễn đạt và bị giới hạn bởi bốn lớp ràng buộc. Về đầu vào, lời nhắc chỉ chứa đề bài, đoạn mã liên quan, gói bằng chứng và mức hỗ trợ cho phép; nội dung do người học nhập được coi là dữ liệu không đáng tin để giảm nguy cơ chèn chỉ dẫn. Về vai trò, mô hình phải trích dẫn bằng chứng cho mỗi nhận định và không viết mã hoàn chỉnh trừ mức L4. Về đầu ra, phản hồi phải tuân theo lược đồ ở Mục 4.3; nếu không hợp lệ sau số lần sinh lại giới hạn, hệ thống dùng phản hồi mẫu theo nhóm lỗi. Cuối cùng, bộ điều phối kiểm tra sau sinh: mọi nhận định có tham chiếu bằng

chứng, không chứa khối mã dài ở mức L1–L3 và không tiết lộ test ẩn hay lời giải chuẩn. Lời giải chuẩn và test ẩn vì vậy chỉ được dùng ở bộ thực thi, không đưa vào lời nhắc.

4.8. An toàn thực thi, quyền riêng tư và vai trò của giáo viên

Mã do người học nộp là mã không đáng tin nên phải chạy trong sandbox có giới hạn thời gian và tài nguyên, hệ thống tệp chỉ đọc, chặn truy cập mạng, danh sách thư viện cho phép và môi trường cô lập mới cho mỗi lần chạy. Các biện pháp này giảm rủi ro nhưng không loại bỏ hoàn toàn, nên cần kiểm thử xâm nhập định kỳ và đánh giá rủi ro liên tục theo tinh thần NIST AI RMF (Tabassi, 2023).

Về dữ liệu, theo hướng dẫn của UNESCO (Miao & Holmes, 2023), hệ thống chỉ nên lưu các trường cần thiết, dùng mã ẩn danh khi gọi mô hình bên ngoài, phân quyền truy cập, đặt thời hạn lưu trữ và thông báo rõ cho người học rằng AI có thể sai. Nhật ký sửa mã là chỉ báo gián tiếp về quá trình học, không nên dùng làm bằng chứng kỷ luật nếu thiếu bối cảnh. Giáo viên giữ vai trò trung tâm: bảng điều khiển cho họ xem phân bố nhóm lỗi của lớp, những sinh viên gặp khó khăn kéo dài và các phản hồi bị chặn; họ cũng cấu hình ngưỡng chuyển mức, bật tắt L4 và đánh dấu phản hồi sai để cải thiện hệ thống.

5. Ví dụ minh họa: phát hiện lỗi trong hàm tính trung bình

Xét bài toán: viết hàm average(a) trả về giá trị trung bình của một danh sách số. Người học viết mã:

```
def average(a):
  return sum(a) / len(a)
```

Với đầu vào [2, 4, 6], chương trình cho kết quả 4.0. Tuy nhiên, test case [] gây lỗi do len(a)=0. Trong hệ thống đề xuất, phản hồi đầu tiên không nhất thiết đưa ngay mã sửa. Hệ thống có thể thông báo test case thất bại, chỉ ra biểu thức gây lỗi và đặt câu hỏi: "Hàm cần trả về gì khi đầu vào là danh sách rỗng?". Người học sau đó có thể lựa chọn một quy ước, chẳng hạn trả về None hoặc phát sinh ngoại lệ có thông báo rõ ràng, tùy yêu cầu bài toán.

Ví dụ cho thấy AI có thể hỗ trợ diễn giải nhanh, nhưng quyết định về hành vi của hàm vẫn thuộc về người học và yêu cầu của bài tập. Sau khi sửa, người học phải chạy lại toàn bộ test case. Nhật ký các lần sửa có thể được dùng để đánh giá quá trình thay vì chỉ đánh giá chương trình cuối cùng.

Bảng 4: Minh họa bộ kiểm thử cho ví dụ

Test case	Kết quả mong đợi	Mục đích	Trạng thái ban đầu
[2,4,6]	4.0	Kiểm tra trường hợp thông thường	Đạt
[1]	1.0	Kiểm tra danh sách 1 phần tử	Đạt
[-2,0,2]	0.0	Kiểm tra số âm và 0	Đạt
[]	Theo quy ước của bài	Kiểm tra trường hợp biên	Lỗi

Diễn tiến một phiên làm bài giả định như sau (chưa phải dữ liệu thực nghiệm). Ở lần nộp đầu, hệ thống gán nhóm lỗi "trường hợp biên" với mức chắc chắn cao nhờ dấu vết ZeroDivisionError và hiển thị phản hồi L1: "Test thứ tư thất bại ở dòng 2." Nếu người học chỉ bọc mã bằng try/except mà không xác định giá trị trả về và số test thất bại không đổi, bộ điều phối nâng lên L2 và yêu cầu họ mô tả hành vi mong muốn trước khi chạy lại. Chỉ sau hai lần nộp không tiến triển, L3 mới giải thích quan hệ nguyên nhân – kết quả; còn L4 chỉ xuất hiện khi người học chủ động yêu cầu, dưới dạng so sánh khác biệt kèm câu hỏi phản tư.

Một ví dụ khác là hàm largest(a) với mã m = 0 rồi duyệt danh sách để cập nhật m. Mã đúng với [3, 9, 2] nhưng trả về 0 với [-5, -2, -9] thay vì -2. Đây là lỗi logic do khởi tạo biến tích lũy bằng 0, không phát sinh ngoại lệ nên thông báo của trình thông dịch không giúp ích; bằng chứng duy nhất là test chứa toàn số âm. Hệ thống hiển thị đầu ra thực tế và mong đợi rồi hỏi: "Điều gì xảy ra nếu mọi phần tử đều nhỏ hơn giá trị khởi tạo?". Khác với ví dụ average, quy ước ở đây rõ ràng (danh sách không rỗng) nên đáp án là duy nhất và có thể chuyển sang L3 sớm hơn; ở average, hệ thống không áp đặt đáp án mà yêu cầu người học làm rõ yêu cầu.

6. Đề xuất thiết kế thực nghiệm

Do chưa có dữ liệu lớp học, bài báo chỉ đề xuất, chưa thực hiện, kế hoạch đánh giá. Giả thuyết chính

là: so với phản hồi chỉ báo đạt/không đạt hoặc trợ lý AI không bị ràng buộc, phản hồi theo mức dựa trên bằng chứng giúp người học tự sửa lỗi thành công nhiều hơn với ít lần xem mẫu hơn, giảm tỷ lệ phản hồi thiếu căn cứ hoặc lộ đáp án, và không làm giảm kết quả ở bài kiểm tra không có hỗ trợ AI. Bảng 4 tóm tắt khung thiết kế đề xuất trong một học phần nhập môn lập trình kéo dài một học kỳ.

Bảng 5: Khung thiết kế thực nghiệm đề xuất

Yếu tố	Mô tả đề xuất
Các nhóm	Đối chứng 1: chấm bài tự động chỉ báo đạt/không đạt; đối chứng 2: trợ lý AI thông thường; thực nghiệm: hệ thống đề xuất L1–L4
Biến phụ thuộc chính	Điểm bài kiểm tra phát hiện và giải thích lỗi không dùng AI; tỷ lệ tự sửa thành công
Biến phụ thuộc phụ	Số lần nộp đến khi đạt; số lần xem mẫu; thời gian sửa lỗi; mức độ hài lòng
Biến kiểm soát	Điểm đầu vào, kinh nghiệm lập trình, giảng viên, độ khó bài tập
Đánh giá chất lượng phản hồi	Rubric năm tiêu chí (đúng, bám bằng chứng, phù hợp trình độ, gợi mở, rõ ràng), hai chuyên gia chấm độc lập và che nhãn nhóm
Mối đe dọa chính	Hiệu ứng mới lạ, sinh viên dùng thêm công cụ AI bên ngoài, khác biệt nền tảng giữa các nhóm, thiên lệch người chấm

Việc thêm nhóm đối chứng thứ hai giúp tách tác động của "có AI" khỏi tác động của "thiết kế sư phạm có kiểm soát". Các mối đe dọa trong khung thiết kế trên được giảm thiểu bằng phân nhóm ngẫu nhiên, hiệu chỉnh theo điểm đầu vào, đo lặp lại ở nhiều thời điểm và đánh giá cuối kỳ có giám sát. Nghiên cứu cần được cơ quan có thẩm quyền về đạo đức nghiên cứu của nhà trường chấp thuận và người tham gia phải được thông báo về dữ liệu thu thập.

7. Bàn luận

Hệ thống đề xuất có thể được triển khai trong các học phần nhập môn lập trình, cấu trúc dữ liệu hoặc phát triển phần mềm. Giá trị chính không nằm ở việc tự động sinh thêm mã, mà ở khả năng tổ chức phản hồi thành một chu trình có bằng chứng. Khi người học nhìn thấy test case thất bại, nguyên nhân và câu hỏi định hướng, họ có cơ sở để tự kiểm tra giả thuyết sửa lỗi.

Một vấn đề quan trọng là nguy cơ người học phụ thuộc vào AI. Nếu hệ thống luôn trả về lời giải hoàn chỉnh, hoạt động nhận thức có thể bị rút ngắn. Do đó, mức phản hồi nên tăng dần theo nhu cầu và lịch sử thao tác. Cách tiếp cận này cũng phù hợp với quan điểm sử dụng AI có trách nhiệm trong giáo dục, trong đó người học vẫn phải chịu trách nhiệm đối với sản phẩm của mình (Miao & Holmes, 2023).

Về kỹ thuật, hệ thống cần kiểm soát môi trường thực thi. Mã do người học gửi có thể chứa vòng lặp vô hạn, thao tác tệp hoặc hành vi không an toàn. NIST AI RMF cung cấp một khung tham chiếu để xem xét rủi ro xuyên suốt vòng đời AI, trong đó các hoạt động quản trị, lập bản đồ, đo lường và quản lý cần được thực hiện liên tục (Tabassi, 2023). Vì vậy, phiên bản triển khai thực tế nên có sandbox, giới hạn thời gian chạy, giới hạn tài nguyên và cơ chế lọc đầu vào.

Về đánh giá, bài báo chưa có dữ liệu lớp học nên chưa thể kết luận hệ thống làm tăng điểm số hay năng lực lập trình. Các kết quả được trình bày chỉ chứng minh tính nhất quán của kiến trúc và khả năng mô tả một kịch bản sử dụng. Đây là điểm cần giữ rõ ràng khi phát triển thành bài báo thực nghiệm.

Nghiên cứu còn một số hạn chế. Ví dụ minh họa dựa trên các bài toán nhỏ bằng Python; bài toán nhiều tệp hoặc có tương tác với cơ sở dữ liệu sẽ đặt ra thách thức mới cho phân tích tĩnh và sandbox. Phân loại lỗi và các ngưỡng chuyển mức mới ở mức khởi đầu. Chất lượng diễn đạt phụ thuộc phiên bản mô hình ngôn ngữ nên cần kiểm tra hồi quy khi cập nhật. Cuối cùng, hệ thống phụ thuộc vào bộ kiểm thử tốt: nếu giáo viên viết test thiếu trường hợp biên thì bằng chứng, và do đó phản hồi, cũng sẽ nghèo nàn. So với các trợ lý được thiết kế để không đưa mã hoàn chỉnh ngay như CodeAid (Kazemitabaar et al., 2024), đề xuất này nhấn mạnh việc ràng buộc nhận định bằng bằng chứng thực thi và mô hình hóa tường minh điều kiện chuyển mức; tuy nhiên đây mới là so sánh ở mức khái niệm.

8. Kết luận

Bài báo đã đề xuất kiến trúc và quy trình cho một hệ thống trợ giảng lập trình dựa trên AI theo định

hướng hỗ trợ người học tự phát hiện và sửa lỗi. Hệ thống kết hợp phân tích cấu trúc mã, thực thi kiểm thử và mô hình ngôn ngữ để tạo phản hồi có căn cứ. Quy trình năm bước gồm xác định mục tiêu, nộp mã và test case, phân tích lỗi, giải thích và gợi ý, sau đó sửa mã và phản tư.

Điểm trọng tâm của mô hình là chuyển AI từ vai trò "sinh lời giải" sang vai trò "hỗ trợ phản hồi". Điều này giúp duy trì hoạt động tự phân tích của người học và tạo dữ liệu về quá trình sửa mã. Tuy nhiên, hiệu quả giáo dục chưa được kiểm chứng thực nghiệm trong nghiên cứu này. Hướng nghiên cứu tiếp theo cần xây dựng bộ dữ liệu lỗi lập trình có gán nhãn, thực nghiệm với nhóm người học theo thiết kế ở Mục 6, đánh giá chất lượng phản hồi bởi chuyên gia và đo lường sự thay đổi về năng lực phát hiện lỗi, giải thích thuật toán và giải quyết vấn đề. Đồng thời, cần đánh giá riêng các rủi ro về bảo mật mã nguồn, quyền riêng tư và sự phụ thuộc vào AI.

Tài liệu tham khảo

Ala-Mutka, K. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education, 15*(2), 83–102. https://doi.org/10.1080/08993400500150747

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4*(2), 167–207. https://doi.org/10.1207/s15327809jls0402_2

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion, 58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research, 77*(1), 81–112. https://doi.org/10.3102/003465430298487

Kazemitabaar, M., Ye, R., Wang, X., Henley, A. Z., Denny, P., Craig, M., & Grossman, T. (2024). CodeAid: Evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery. https://doi.org/10.1145/3613904.3642773

Keuning, H., Jeuring, J., & Heeren, B. (2019). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education, 19*(1), Article 3. https://doi.org/10.1145/3231711

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30* (pp. 4765–4774). Curran Associates.

Miao, F., & Holmes, W. (2023). *Guidance for generative AI in education and research*. UNESCO. https://doi.org/10.54675/EWZM9535

Molnar, C. (2022). *Interpretable machine learning* (2nd ed.). https://christophm.github.io/interpretable-ml-book/

Prather, J., Denny, P., Leinonen, J., Becker, B. A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Peterson, A., Pettit, R., Reeves, B. N., & Savelka, J. (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*. Association for Computing Machinery. https://doi.org/10.1145/3623762.3633499

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Tabassi, E. (2023). *Artificial intelligence risk management framework (AI RMF 1.0)* (NIST AI 100-1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist, 46*(4), 197–221. https://doi.org/10.1080/00461520.2011.611369

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes* (M. Cole, V. John-Steiner, S. Scribner, & E. Souberman, Eds.). Harvard University Press.

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry, 17*(2), 89–100. https://doi.org/10.1111/j.1469-7610.1976.tb00381.x