# ỨNG DỤNG VIBE CODING TRONG GIẢNG DẠY LẬP TRÌNH TẠI CÁC TRƯỜNG ĐẠI HỌC

**NGUYỄN PHỒN LỮA**
**Trường Đại học Công nghệ Đông Á**

*Nhận bài ngày 21/12/2025. Sửa chữa xong 21/01/2026. Duyệt đăng 27/01/2026.*

***Abstract***

*The rapid advancement of large language models (LLMs) has introduced Vibe Coding, a paradigm in which programmers express intent in natural language and AI systems generate code accordingly. This paper reviews the concept of Vibe Coding, analyzes its pedagogical benefits and implementation challenges, and proposes a hybrid teaching model for integrating Vibe Coding into programming curricula. The proposed approach aims to help students develop both foundational programming competence and practical capabilities for effective collaboration with AI tools.*

***Keywords:*** *AI-assisted programming, large language models, prompt engineering, Vibe Coding.*

## 1. Đặt vấn đề

Trong hơn nửa thế kỷ qua, phương pháp giảng dạy lập trình đã trải qua nhiều giai đoạn phát triển nhưng vẫn được thực hiện một mô hình truyền thống: giảng viên hướng dẫn cú pháp, cấu trúc dữ liệu và thuật toán; sinh viên (SV) thực hành bằng cách viết mã trực tiếp trên các trình biên dịch hoặc môi trường phát triển tích hợp (IDE). Mô hình này đã chứng minh hiệu quả trong việc xây dựng nền tảng tư duy logic và kỹ năng giải quyết vấn đề cho hàng triệu lập trình viên trên toàn thế giới [6], [7].

Tuy nhiên, trong bối cảnh công nghệ đang thay đổi với tốc độ chưa từng có, với sự xuất hiện và phát triển vượt bậc của trí tuệ nhân tạo (AI), đặc biệt là các mô hình ngôn ngữ lớn (Large Language Models - LLMs) như GPT-4, Claude, Gemini, và các công cụ lập trình hỗ trợ AI như GitHub Copilot [2], ChatGPT Code Interpreter, Replit AI, đã tạo nên một cuộc cách mạng trong cách thức con người tương tác với máy tính và phát triển phần mềm.

Năm 2024 đánh dấu một bước ngoặt quan trọng khi khái niệm "Vibe Coding" được Andrej Karpathy - một trong những nhà nghiên cứu AI hàng đầu thế giới, cựu Giám đốc AI tại Tesla - giới thiệu rộng rãi [4]. Thuật ngữ này mô tả một phong cách lập trình hoàn toàn mới: "viết phần mềm thông qua hội thoại và cảm nhận, thay vì thao tác mã hóa chi tiết từng dòng" [4]. Theo cách này, lập trình viên không còn tập trung vào việc viết code, mà chuyển sang vai trò là người thiết kế, người định hướng - mô tả mục tiêu, hành vi mong muốn, hoặc "vibe" (cảm nhận tổng thể) của ứng dụng, và để hệ thống AI đảm nhiệm việc sinh mã, kiểm thử, và hoàn thiện chương trình [5].

Sự chuyển dịch sang Vibe Coding không chỉ là thay đổi về phương thức mà còn tạo ra bước nhảy vọt về hiệu suất lao động. Nghiên cứu thực nghiệm của Ziegler và cộng sự chỉ ra rằng các công cụ hỗ trợ AI giúp lập trình viên hoàn thành nhiệm vụ nhanh hơn tới 55,8% so với phương pháp thủ công, đồng thời tăng đáng kể tỷ lệ hoàn thành dự án thành công [10]. Thực tế này tương đồng với dữ liệu từ Stack Overflow, khi phần lớn lập trình viên thừa nhận rằng việc sử dụng AI không chỉ giúp họ giải quyết các tác vụ lặp đi lặp lại mà còn cho phép họ dành nhiều thời gian hơn cho các công việc có giá trị cao như thiết kế kiến trúc và tối ưu hóa trải nghiệm người dùng [8].

*Email: luanp@eaut.edu.vn*

## 2. Nội dung nghiên cứu

### *2.1. Giới thiệu về Vibe Coding và sự tiến hóa của lập trình*

*2.1.1. Giới thiệu Vibe Coding*

Vibe Coding là một phương thức phát triển phần mềm mới, trong đó con người giao tiếp với hệ thống trí tuệ nhân tạo thông qua ngôn ngữ tự nhiên để tạo, chỉnh sửa, kiểm thử và triển khai mã nguồn. Thuật ngữ "vibe" nhấn mạnh tính chất trực quan và định hướng trải nghiệm; Karpathy khẳng định Vibe Coding không tập trung vào sự chính xác tuyệt đối ngay từ đầu mà chú trọng vào "cảm nhận" (vibe) về kết quả đầu ra [4]. Khác với việc sinh mã (code generation) đơn thuần, Vibe Coding là quá trình đối thoại liên tục để tinh chỉnh sản phẩm dựa trên mô tả mục tiêu và hành vi mong muốn.

Vibe Coding trở thành hiện thực nhờ sự kết hợp của nhiều công nghệ trí tuệ nhân tạo tiên tiến:

(1) Mô hình ngôn ngữ lớn (LLMs): Các mô hình như GPT-4 , Claude và Gemini có khả năng hiểu ngữ cảnh, sinh mã nguồn đa ngôn ngữ và giải thích các cấu trúc phức tạp.

(2) Công cụ lập trình hỗ trợ AI: Các nền tảng như GitHub Copilot hay Cursor tích hợp trực tiếp AI vào IDE, giúp hoàn thiện hàm và thực hiện refactoring hiệu quả.

(3) Kỹ thuật Prompt Engineering: Chất lượng đầu ra phụ thuộc vào việc thiết kế câu lệnh rõ ràng và cung cấp ví dụ minh họa (few-shot learning) để AI hiểu đúng ý định người dùng.

Theo Kim & Yegge, chu trình Vibe Coding tối ưu hóa sự tương tác giữa người và máy bao gồm bảy bước bảy bước sau đây [5]:

(1) Xác định mục tiêu rõ ràng (Frame your objective): Người phát triển cung cấp cho AI một tổng quan cụ thể, súc tích về kết quả mong muốn, bao gồm xác định rõ tiêu chí thành công và lý do xây dựng dự án.

(2) Phân tách nhiệm vụ (Decompose the tasks): Các công việc phức tạp cần được chia nhỏ thành các bước có thể thực hiện được. Càng tách nhỏ nhiệm vụ, AI càng có khả năng thực hiện thành công cao hơn.

(3) Khởi động quá trình tương tác (Start the conversation): Người dùng đưa ra yêu cầu hoặc hướng dẫn cụ thể để AI xây dựng kế hoạch hoặc bắt đầu thực hiện các bước cần thiết.

(4) Đánh giá kỹ lưỡng sản phẩm (Review with care): Mặc dù kết quả do AI cung cấp có thể trông chính xác, người dùng vẫn cần kiểm tra và đánh giá lại để thiết lập độ tin cậy, tránh những sai sót tiềm ẩn.

(5) Kiểm thử và xác nhận (Test and verify): Chất lượng mã nguồn là trách nhiệm của người phát triển. Việc chuẩn bị các bài kiểm thử trước khi tạo mã cho phép phát hiện nhanh lỗi, sửa chữa kịp thời.

(6) Tinh chỉnh và lặp lại (Refine and iterate): Quá trình phát triển cần được tiếp tục cải tiến dựa trên kết quả đánh giá và thử nghiệm.

(7) Tự động hóa quy trình (Automate your own workflow): Người phát triển thành thạo có thể tự động hóa quy trình giúp giảm thiểu các công việc thủ công làm chậm trễ quá trình Vibe Coding.

*2.1.2. Ưu, nhược điểm của Vibe Coding*

Vibe Coding kết hợp AI giúp rút ngắn đáng kể chu trình phát triển phần mềm, đưa ý tưởng thành nguyên mẫu chỉ trong vài giờ thay vì vài ngày. Mô hình này mở rộng khả năng lập trình cho cả người không chuyên như nhà thiết kế, quản lý sản phẩm hay doanh nhân, thúc đẩy sáng tạo và thử nghiệm nhanh các giải pháp mới. Đồng thời, AI hỗ trợ học tập hiệu quả, giúp người học nắm rõ logic, best practices và tiếp cận lập trình dễ dàng hơn. Code do AI sinh ra thường có cấu trúc tốt, dễ bảo trì và kèm tài liệu chú thích đầy đủ.

Mặc dù có những ưu điểm dễ thấy trên đây nhưng Vibe Coding cũng có những nhược điểm đáng kể. Cụ thể là kỹ thuật này phụ thuộc mạnh vào độ chính xác của AI, dễ phát sinh lỗi logic và khó xử lý ngữ cảnh (context) hoặc trường hợp biên (edge case) phức tạp. Chất lượng đầu ra phụ thuộc vào kỹ năng viết prompt của người dùng. Sự lệ thuộc quá mức vào AI có thể làm suy giảm kỹ năng lập trình, debug và tối ưu hóa. Ngoài ra, còn tồn tại rủi ro về bảo mật, bản quyền, quyền riêng tư dữ liệu và khó đảm bảo kiểm thử, bảo trì khi hệ thống mở rộng. Việc thiếu minh bạch trong cách AI sinh mã cũng gây lo ngại cho các ứng dụng đòi hỏi độ tin cậy cao.

***2.2. Một số vấn đề đặt ra trong việc ứng dụng Vibe Coding trong giảng dạy lập trình tại các trường đại học hiện nay***

*2.2.1. Thực trạng giảng dạy lập trình hiện nay*

Chương trình giảng dạy lập trình ở các trường đại học Việt Nam hiện nay chủ yếu dựa trên mô hình truyền thống đã tồn tại trong nhiều thập kỷ.

Cấu trúc chương trình học: SV ngành Công nghệ Thông tin thường được đào tạo theo trình tự: Năm nhất tập trung vào nhập môn lập trình (C/C++) và cấu trúc dữ liệu cùng giải thuật cơ bản; năm hai giảng dạy lập trình hướng đối tượng (Java/C#), cơ sở dữ liệu và kiến thức web cơ bản; năm ba phát triển ứng dụng web/mobile, công nghệ phần mềm (Software Engineering) và thiết kế hệ thống; năm cuối là thực hiện đồ án tốt nghiệp và chuyên sâu theo chuyên ngành.

Phương pháp giảng dạy chủ đạo: Phương pháp chủ yếu gồm việc giảng viên trình bày lý thuyết về cú pháp và các khái niệm lập trình, minh họa bằng demo code trên lớp, sau đó SV thực hành qua các bài tập nhỏ theo mẫu có sẵn. Đánh giá thường dựa trên bài kiểm tra hoặc các assignment cá nhân, trong khi đồ án cuối kỳ phần lớn là sao chép các ứng dụng đơn giản.

Hạn chế của mô hình hiện tại: Chương trình giảng dạy lập trình tại các trường đại học hiện nay chủ yếu dựa trên mô hình truyền thống: tập trung vào cú pháp ngôn ngữ, cấu trúc dữ liệu và giải thuật cơ bản theo trình tự tuyến tính [7]. Tuy nhiên, mô hình này đang bộc lộ nhiều hạn chế, đặc biệt là việc tập trung quá mức vào sửa lỗi cú pháp khiến SV thiếu cơ hội phát triển tư duy thiết kế hệ thống và chưa được chuẩn bị đầy đủ kỹ năng làm việc với các công cụ trí tuệ nhân tạo đang thay đổi từng ngày.

*2.2.2. Sự chuyển dịch trong ngành công nghiệp phần mềm và yêu cầu năng lực cho lập trình viên tương lai*

Ngành công nghiệp phần mềm đang trải qua những chuyển biến sâu sắc. Theo khảo sát của Stack Overflow (2024), có đến 76% lập trình viên chuyên nghiệp đã sử dụng hoặc dự kiến sử dụng các công cụ hỗ trợ bởi AI và 44% doanh nghiệp đã chính thức tích hợp AI vào quy trình sản xuất [8]. Sự thay đổi này không chỉ dừng lại ở việc sinh mã mà còn mở rộng sang tự động hóa kiểm thử, code review và tối ưu hóa hệ thống, giúp gia tăng đáng kể năng suất lao động [10].

Để thích ứng, SV cần được trang bị nhóm năng lực đa chiều. Bên cạnh kỹ năng nền tảng về thuật toán và tư duy logic [6], SV phải làm chủ kỹ năng tương tác với AI (AI Collaboration Skills). Điều này bao gồm khả năng thiết kế prompt hiệu quả và năng lực đánh giá, thẩm định mã nguồn do AI sinh ra để tránh những sai sót về logic hoặc bảo mật [1], [10]. Tư duy thiết kế hệ thống và khả năng học hỏi liên tục trở thành yếu tố then chốt thay vì chỉ thuần thục kỹ năng viết mã thủ công.

***2.3. Một số giải pháp nâng cao hiệu quả ứng dụng Vibe Coding trong giảng dạy lập trình tại các trường đại học hiện nay***

*2.3.1. Xây dựng nguyên tắc thiết kế mô hình giảng dạy*

Mô hình giảng dạy được đề xuất dựa trên những nguyên tắc cơ bản nhằm đảm bảo sự cân bằng và hiệu quả:

(1) Cân bằng: kết hợp hài hòa truyền thống và AI.

(2) Tuần tự: học nền tảng trước, AI sau.

(3) Thực hành: Project-based learning, giảm lý thuyết khô khan.

(4) Phản tư: phân tích, so sánh mã tự viết và AI.

(5) Thích ứng: linh hoạt với công nghệ AI.

*2.3.2. Triển khai mô hình giảng dạy kết hợp bốn giai đoạn*

Dưới đây là đề xuất mô hình ứng dụng Vibe Coding trong giảng dạy lập trình bốn giai đoạn tương ứng với chương trình đào tạo bốn năm.

Giai đoạn 1: Xây dựng kiến thức nền tảng (Foundation Phase)

Giai đoạn này nhằm hình thành nền tảng tư duy tính toán, khả năng giải quyết vấn đề thuật toán và kỹ năng lập trình căn bản thông qua việc viết, đọc và gỡ lỗi mã nguồn thủ công.

Nội dung học tập bao gồm lập trình cơ bản (C/C++) với cấu trúc ngôn ngữ, kiểu dữ liệu, hàm, con trỏ; cấu trúc dữ liệu và giải thuật; lập trình hướng đối tượng (Java/C#); phân tích độ phức tạp thuật toán với Big O. Phương pháp giảng dạy kết hợp giảng lý thuyết (30%), trình diễn mã (20%), thực hành hướng dẫn (30%), tự học (20%). Môi trường học tập sử dụng IDE truyền thống và hệ thống chấm tự động.

Đánh giá qua bài tập tuần (30%), kiểm tra giữa/cuối kỳ (50%), đồ án nhỏ (20%). SV không được phép sử dụng công cụ AI nhằm đảm bảo phát triển năng lực nền tảng chân thực.

Giai đoạn 2: Tương tác với AI (AI Collaboration Phase)

Giai đoạn này giúp SV làm quen và khai thác hiệu quả công cụ AI, hiểu cơ chế, giới hạn, tác động. SV học viết prompt hiệu quả, đánh giá chất lượng mã AI, phát triển tư duy phản biện.

Nội dung học tập gồm giới thiệu AI và LLMs (ChatGPT, Copilot, Claude); kỹ thuật Prompt Engineering với xây dựng prompt rõ ràng, few-shot, role-based, iterative prompting; đánh giá và tối ưu mã AI; đạo đức AI về bản quyền, bảo mật, minh bạch. Phương pháp giảng dạy chú trọng workshop thực hành (50%), phân tích tình huống (20%), thảo luận nhóm (15%), giảng lý thuyết (15%). Công cụ sử dụng: Cursor IDE, VS Code với Copilot, Replit AI, GitHub.

Đánh giá qua bài tập Prompt Engineering (20%), phân tích mã AI (20%), dự án tích hợp AI (30%), kiểm tra cuối kỳ (20%), thảo luận (10%).

Giai đoạn 3: Thực hành dự án Vibe Coding (Project-Based Phase)

Giai đoạn này giúp SV áp dụng Vibe Coding phát triển sản phẩm hoàn chỉnh. SV rèn kỹ năng nhóm, thiết kế hệ thống, quản lý dự án, dùng AI như "thành viên ảo".

Nội dung học tập bao gồm khởi tạo dự án với nhóm 3-4 người, chọn đề tài, thiết kế kiến trúc; phát triển theo Sprint với AI sinh code, gỡ lỗi, viết test; triển khai sản phẩm qua DevOps (Vercel, Netlify, Railway); phản hồi qua sprint review. Phương pháp giảng dạy theo project-based learning với SV tự chủ; mentoring cá nhân và nhóm; phản hồi liên tục. Sử dụng công cụ chuyên nghiệp: React/Vue, Node.js/Python, GitHub, Trello, Figma, Slack.

Đánh giá theo sprint (40%) dựa trên mã nguồn, kiểm thử, tài liệu và hợp tác; dự án cuối kỳ (40%) chấm theo chất lượng sản phẩm, tính sáng tạo; báo cáo cá nhân (20%) mô tả vai trò, quá trình sử dụng AI.

Giai đoạn 4: Phân tích và phản tư (Analysis & Reflection Phase)

Giai đoạn cuối hướng đến phát triển tư duy phản biện, khả năng phân tích sâu về vai trò, giới hạn và ảnh hưởng của AI. SV hình thành quan điểm cá nhân, định hướng nghề nghiệp và chiến lược học tập dài hạn.

Nội dung học tập gồm phân tích và so sánh kết quả giữa lập trình truyền thống và có AI qua workshop và thử thách thuật toán; nghiên cứu tình huống phân tích thành công và thất bại của AI trong doanh nghiệp; seminar và tranh biện với chuyên gia về xu hướng nghề, đạo đức AI; nghiên cứu cá nhân thực hiện bài báo khoa học (5.000-6.000 từ). Phương pháp giảng dạy kết hợp workshop phân tích (30%), thảo luận và seminar (30%), nghiên cứu độc lập (30%), phản tư cá nhân (10%).

Đánh giá qua tham gia workshop và thảo luận (20%), phân tích tình huống (15%), tranh biện học thuật (10%), bài nghiên cứu (35%), thuyết trình và kế hoạch phát triển (20%).

*2.3.3. Nâng cao năng lực sử dụng công nghệ cho giảng viên và sinh viên*

Một trong những điều kiện tiên quyết quyết định hiệu quả ứng dụng Vibe Coding là nâng cao năng lực sử dụng công nghệ cho cả giảng viên và SV. Đối với giảng viên, việc nâng cao năng lực công nghệ không chỉ dừng lại ở khả năng sử dụng các tính năng cơ bản mà còn phải hướng đến thiết kế hoạt động học tập trực tuyến, xây dựng ngân hàng học liệu, quản lý lớp học, giao nhiệm vụ, chấm điểm và phản hồi trên nền tảng số. Các trường cần tổ chức bồi dưỡng thường xuyên, xây dựng chương trình tập huấn theo từng trình độ và nhu cầu.

Đối với SV, việc nâng cao kỹ năng sử dụng công nghệ cần được thực hiện ngay từ năm đầu, thông qua các khóa bồi dưỡng về kỹ năng sử dụng công cụ AI, kỹ năng học tập với AI, kỹ năng quản lý tiến độ, nộp bài, trao đổi học thuật. Việc này giúp SV hình thành thói quen chủ động trong học tập, biết khai thác tài nguyên học tập, theo dõi tiến độ, qua đó nâng cao năng lực tự học và tự đánh giá.

*2.3.4. Hoàn thiện hạ tầng công nghệ và xây dựng cơ chế đánh giá, phản hồi*

Để Vibe Coding phát huy tối đa hiệu quả, các trường cần hoàn thiện hạ tầng công nghệ, nâng cấp đường truyền mạng, máy chủ và trung tâm dữ liệu, bảo đảm khả năng xử lý lượng truy cập lớn. Việc áp dụng đồng bộ các tiêu chuẩn như SCORM, LTI hay các nền tảng Cloud hiện đại sẽ giúp hệ thống vận hành ổn định, dễ nâng cấp và bảo trì.

Bên cạnh đó, cần thiết lập công cụ và tiêu chí đánh giá hiệu quả theo từng giai đoạn, bao gồm mức độ truy cập, tần suất sử dụng, khả năng hỗ trợ giảng dạy, độ ổn định kỹ thuật. Hệ thống phản hồi từ người sử dụng cần được xây dựng đầy đủ, thông suốt, giúp SV và giảng viên dễ dàng chia sẻ ý kiến, phản ánh lỗi kỹ thuật hoặc đề xuất tính năng cải tiến. Nhà trường cần có cơ chế tiếp nhận, xử lý và công bố kết quả phản hồi một cách minh bạch, tạo dựng niềm tin và khuyến khích sự tham gia tích cực.

**3. Kết luận**

Ứng dụng Vibe Coding trong giảng dạy lập trình tại các trường đại học là yêu cầu tất yếu trong bối cảnh đổi mới giáo dục và chuyển đổi số hiện nay. Vibe Coding không chỉ giúp tối ưu hóa quá trình dạy học, mà còn tạo môi trường học tập mở, linh hoạt, cho phép SV chủ động phát triển năng lực theo yêu cầu của xã hội hiện đại. Tuy nhiên, để khai thác hết tiềm năng các trường cần thực hiện đồng bộ nhiều giải pháp.

Mô hình giảng dạy kết hợp bốn giai đoạn được đề xuất trong nghiên cứu này giúp cân bằng giữa kỹ năng lập trình truyền thống và khả năng cộng tác với AI. Giai đoạn xây dựng nền tảng đảm bảo SV có tư duy máy tính cốt lõi; giai đoạn tương tác với AI giúp làm quen công cụ và kỹ thuật Prompt Engineering; giai đoạn thực hành dự án áp dụng Vibe Coding vào phát triển sản phẩm thực tế; giai đoạn phân tích và phản tư phát triển tư duy phản biện về vai trò AI.

Để triển khai hiệu quả mô hình này, các trường cần chú trọng hoàn thiện hạ tầng công nghệ, nâng cao năng lực số cho giảng viên và SV, đồng thời xây dựng cơ chế phản hồi, cải tiến thường xuyên. Trên cơ sở đó, có thể triển khai các giải pháp hỗ trợ kịp thời, góp phần nâng cao chất lượng đào tạo, cung cấp nguồn nhân lực chất lượng cao sẵn sàng làm việc với công nghệ AI, đáp ứng yêu cầu phát triển trong thời kỳ mới.

**Tài liệu tham khảo**


[1] Barke, S., James, M. B., & Polikarpova, N. (2023). *Grounded Copilot: How Programmers Interact with Code-Generating Models*. Proceedings of the ACM on Programming Languages, 7(OOPSLA1), 85-111.

[2] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., … & Zaremba, W. (2021). *Evaluating Large Language Models Trained on Code.* arXiv preprint arXiv:2107.03374.

[3] Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Professional.

[4] Karpathy, A. (2025). *The Rise of Vibe Coding: Programming in the Age of LLMs.* AI and Software Development Conference, San Francisco.

[5] Kim, G., & Yegge, S. (2025). *Vibe coding: Building production-grade software with GenAI, chat, agents, and beyond.* IT Revolution.

[6] Knuth, D. E. (1997). *The Art of Computer Programming,* Volumes 1-3. Addison-Wesley Professional.

[7] Robins, A., Rountree, J., & Rountree, N. (2003). *Learning and Teaching Programming: A Review and Discussion.* Computer Science Education, 13(2), 137-172.

[8] Stack Overflow. (2024). *2024 Developer Survey Results.* Retrieved from https://survey.stackoverflow.co.

[9] Vaithilingam, P., Zhang, T., & Glassman, E. L. (2022). *Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models*. CHI Conference on Human Factors in Computing Systems Extended Abstracts, pp. 1-7.

[10] Ziegler, A., Kalliamvakou, E., Li, X. A., Rice, A., Rifkin, D., Simister, S., … & Aftandilian, E. (2022). *Productivity Assessment of Neural Code Completion.* Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, pp. 21-29.