VIBE CODING TRONG PHÁT TRIỂN PHẦN MỀM: MỘT TỔNG QUAN HỆ THỐNG VỀ TÁC ĐỘNG ĐẾN TỐC ĐỘ, CHẤT LƯỢNG VÀ NỢ KỸ THUẬT

Quách Thị Hạnh[1*], Nguyễn Tố Uyên[1], Vũ Tất Điệp[1]
[*]Tác giả liên hệ, Email: hanhqt.dttt@hou.edu.vn

Ngày tòa soạn nhận được bài báo: 02/06/2025
Ngày phản biện đánh giá: 08/09/2025
Ngày bài báo được duyệt đăng: 15/10/2025

DOI: 10.59266/houjs.2025.726

Tóm tắt: *Sự phát triển nhanh chóng của mô hình ngôn ngữ lớn (LLM) đã thúc đẩy sự xuất hiện của "vibe coding" - phương pháp lập trình dựa trên mô tả yêu cầu bằng ngôn ngữ tự nhiên để AI sinh mã. Bài báo này thực hiện tổng quan hệ thống (Systematic Literature Review) nhằm tổng hợp các bằng chứng thực nghiệm mới nhất về ảnh hưởng của vibe coding tới tốc độ và chất lượng phát triển phần mềm. Quy trình sàng lọc và phân tích tuân thủ hướng dẫn PRISMA 2020, với dữ liệu từ 17 nghiên cứu công bố trong giai đoạn 2022 - 2025. Kết quả cho thấy vibe coding giúp cải thiện tốc độ phát triển trung bình từ 19% đến 23%. Tuy nhiên, độ chính xác của mã sinh ra chỉ đạt 48%, và tỉ lệ lỗi hoặc lỗ hổng bảo mật trong lần sinh đầu tiên ở mức 31%, phản ánh những rủi ro đáng kể về chất lượng. Dù vậy, khả năng sửa lỗi của hệ thống có thể đạt tới 89% khi áp dụng chiến lược tương tác nhiều vòng hoặc prompt giàu ngữ cảnh. Nghiên cứu làm rõ mối quan hệ giữa tốc độ, chất lượng và bảo mật trong vibe coding, đồng thời nhấn mạnh nhu cầu xây dựng mô hình cộng tác con người - AI hiệu quả hơn nhằm giảm thiểu nợ kỹ thuật và đảm bảo độ tin cậy của phần mềm.*

Từ khóa: *vibe coding, mô hình ngôn ngữ lớn, GitHub Copilot, năng suất lập trình, chất lượng phần mềm*

I. Đặt vấn đề

Sự xuất hiện của các mô hình ngôn ngữ lớn (Large Language Model, LLM) có khả năng sinh mã, tiêu biểu là Codex, GPT- 4 hay AlphaCode, đã mở ra viễn cảnh trong đó lập trình viên chuyển từ vai trò "người viết" sang "người dẫn dắt" quá trình tạo mã (Li và c.s., 2022)yet so far incorporating innovations in AI has proven challenging. Recent large-scale language models have demonstrated an impressive ability to generate code, and are now able

[1] Trường Đại học Mở Hà Nội

to complete simple programming tasks. However, these models still perform poorly when evaluated on more complex, unseen problems that require problem-solving skills beyond simply translating instructions into code. For example, competitive programming problems which require an understanding of algorithms and complex natural language remain extremely challenging. To address this gap, we introduce AlphaCode, a system for code generation that can create novel solutions to these problems that require deeper reasoning. In simulated evaluations on recent programming competitions on the Codeforces platform, AlphaCode achieved on average a ranking of top 54.3% in competitions with more than 5,000 participants. We found that three key components were critical to achieve good and reliable performance: (1. Khái niệm *vibe coding* được Karpathy nêu ra năm 2025 để mô tả trải nghiệm "nói cho máy hiểu" thay vì tự gõ từng dòng lệnh, và nhanh chóng thu hút sự quan tâm của cả cộng đồng kỹ thuật lẫn người dùng không chuyên (Andrej Karpathy [@ karpathy], 2025; Roose, 2025). Các khảo sát ban đầu cho thấy lập trình viên chấp nhận trung bình 27% gợi ý của GitHub Copilot và ước tính tiết kiệm khoảng 20% thời gian phát triển (Ziegler và c.s., 2024). Thực nghiệm đối chứng quy mô 95 lập trình viên ghi nhận nhóm sử dụng Copilot hoàn thành tác vụ nhanh gấp rưỡi nhóm đối chứng mà không làm giảm tỷ lệ hoàn thành (Peng và c.s., 2023).

Tuy vậy, lợi ích về tốc độ không đi kèm bảo đảm chất lượng tuyệt đối. Khoảng một phần ba đoạn mã Python và JavaScript do Copilot sinh trong các kho GitHub công khai chứa lỗ hổng bảo mật thuộc nhóm CWE quan trọng (Fu và c.s., 2025), và Copilot cũng có xu hướng lặp lại lỗi đã biết trong dữ liệu huấn luyện (Asare và c.s., 2024). Ngoài ra, nghiên cứu của Vaithilingam et al. (Vaithilingam và c.s., 2022) cho thấy người dùng Copilot phải tốn nhiều thời gian để hiểu và sửa các lỗi trong mã được tạo. Trong một nghiên cứu khác, Barke et al. (Barke và c.s., 2022) đã phân tách hành vi sử dụng Copilot thành hai hạng mục: "Tăng Tốc", khi người dùng biết các bước cần làm tiếp theo và sử dụng Copilot để tạo mã nhanh hơn, và "Khám Phá", khi người dùng cần Copilot đưa ra các gợi ý và lựa chọn các bước phát triển kế tiếp. Kết quả của thí nghiệm này chỉ ra rằng hiệu quả của vibe coding phụ thuộc rất lớn vào bối cảnh sử dụng và khả năng lập trình của người dùng.

Khoảng trống nghiên cứu hiện nay nằm ở việc thiếu những đánh giá dài hạn về duy trì mã, kiểm thử an toàn và tác động đến quy trình phát triển phần mềm ở quy mô doanh nghiệp. Do đó, bài báo này nhằm (i) tổng hợp có hệ thống các bằng chứng thực nghiệm mới nhất về ảnh hưởng của vibe coding tới tốc độ và chất lượng phát triển phần mềm, (ii) phân tích các mối nguy tiềm ẩn liên quan đến bảo mật và nợ kỹ thuật, và (iii) định hướng nghiên cứu tương lai về mô hình cộng tác con người - AI. Nội dung tiếp theo trình bày chi tiết phương pháp PRISMA, dữ

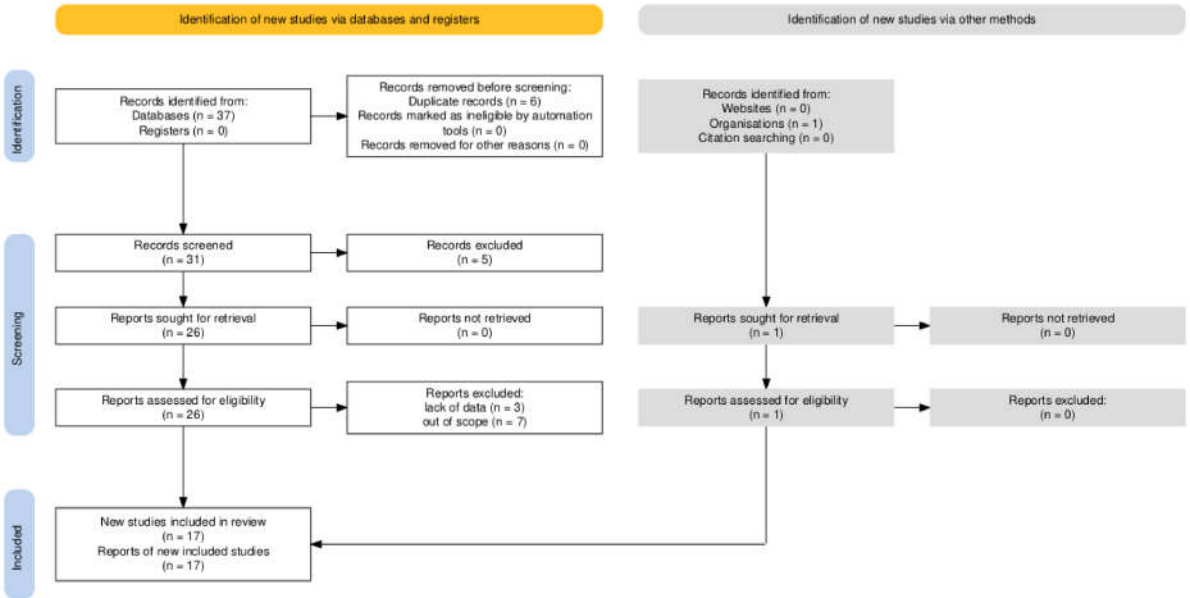
liệu nghiên cứu, kết quả phân tích và thảo luận hàm ý thực tiễn.

II. Phương pháp nghiên cứu

2.1. Tìm kiếm và sàng lọc dữ liệu

Bài tổng quan được tiến hành theo chuẩn PRISMA 2020 nhằm bảo đảm tính minh bạch và khả năng tái lập (Page và c.s., 2021). Hoạt động truy vấn được triển khai trên năm cơ sở dữ liệu học thuật - ACM Digital Library, IEEE Xplore, SpringerLink, ScienceDirect và arXiv - kết hợp tìm kiếm nguồn xám qua Google Scholar, GitHub, Google AI Blog và DeepMind. Khung thời gian thu thập tài liệu giới hạn từ 1/1/2022 đến 31/5/2025; ngôn ngữ chấp nhận gồm tiếng Việt và tiếng Anh. Tổ hợp từ khóa "vibe coding", "natural language programming", "code generation", "productivity" và "software quality" được áp dụng trên trường tiêu đề, tóm tắt cũng như từ khóa của từng cơ sở dữ liệu. Kết quả ban đầu ghi nhận 37 bản ghi.

Tiêu chí đưa vào yêu cầu tài liệu (i) báo cáo dữ liệu định lượng hoặc định tính về tác động của công cụ sinh mã dựa trên mô hình ngôn ngữ lớn đối với tốc độ hoặc chất lượng phát triển phần mềm, (ii) công bố trong giai đoạn khảo sát, (iii) có toàn văn truy cập. Sau khi loại 6 bản ghi trùng lặp, 31 bản ghi được sàng lọc tiêu đề và tóm tắt; 5 bản ghi bị loại vì không đáp ứng tiêu chí nội dung. Giai đoạn truy xuất toàn văn tiếp nhận 26 báo cáo, đồng thời bổ sung 1 báo cáo do tổ chức công bố ngoài hệ thống cơ sở dữ liệu, nâng tổng số báo cáo đánh giá mức độ phù hợp lên 27. Trong số đó, 10 báo cáo bị loại - 3 thiếu dữ liệu thực nghiệm và 7 nằm ngoài phạm vi chủ đề - kết quả còn lại 17 nghiên cứu đáp ứng đầy đủ yêu cầu và được đưa vào phân tích cuối cùng. Toàn bộ quy trình cùng số liệu chi tiết được minh họa tại Hình 1.



Hình 1. Sơ đồ luồng PRISMA

2.2. Trích xuất dữ liệu

Toàn văn của mười bảy nghiên cứu được đọc kỹ và trích các trường thông tin chuẩn hóa gồm: tác giả - năm, loại tài liệu, bối cảnh nghiên cứu, cỡ mẫu/quy mô, công cụ AI sinh mã, ngôn ngữ lập trình, chỉ số tốc độ (thời gian hoàn thành, tỉ lệ gợi ý chấp nhận), chỉ số chất lượng (độ chính xác, lỗ hổng bảo mật) và kết quả chính. Các giá trị được ghi trực tiếp vào bảng tính Excel duy nhất.

2.3. Đánh giá chất lượng và rủi ro thiên lệch

Chất lượng 15 bài báo học thuật được thẩm định bằng Mixed Methods Appraisal Tool 2018 (MMAT), mỗi bài chấm năm tiêu chí; tổng điểm quy đổi thành ba mức cao, trung bình và thấp (Hong và c.s., 2019). Báo cáo nội bộ của Google và white paper của Bakal et al. (Bakal và c.s., 2025) được đánh giá theo khung AACODS nhằm bảo đảm tính uy tín, độ chính xác, phạm vi và tính khách quan (Tyndall, 2010). Kết quả thẩm định được thêm vào bảng tính Excel trích xuất dữ liệu (xem Phụ lục 1).

2.4. Phương pháp tổng hợp

Khi ít nhất ba nghiên cứu sử dụng cùng thước đo định lượng, giá trị trung bình có trọng số được tính và kèm khoảng tin cậy 95%; thống kê thực hiện bằng Python 3.11 với thư viện pandas và statsmodels. Những kết quả không đồng nhất về thiết kế hoặc biến đo lường được tổng hợp tường thuật, nêu rõ xu hướng chung, khoảng trống nghiên cứu và hàm ý thực tiễn. Phân tích nhạy lần lượt loại bỏ những nghiên cứu xếp hạng chất lượng thấp để kiểm tra độ bền vững của kết luận. Các bước trên bảo đảm tuân thủ khuyến nghị báo cáo PRISMA 2020 (Page và c.s., 2021).

III. Kết quả

Kết quả thu được từ phương pháp tổng hợp nêu trên đã chỉ ra tác động của vibe coding trong ba khía cạnh: tốc độ phát triển phần mềm, chất lượng phần mềm, và bảo mật nợ và kỹ thuật. Sau đây là phân tích chi tiết cho từng hạng mục.

3.1. Tốc độ phát triển phần mềm

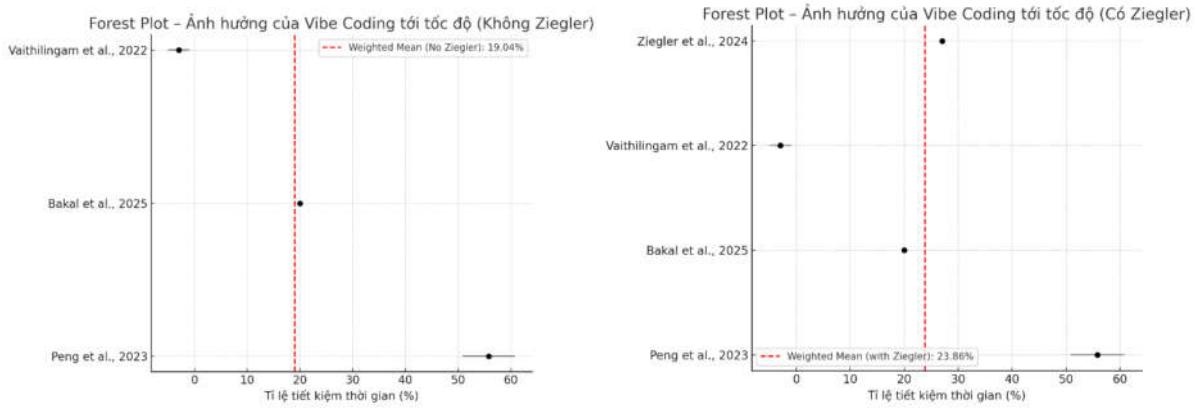
Bốn nghiên cứu cung cấp dữ liệu liên quan đến tác động của vibe coding đối với tốc độ phát triển phần mềm, trong đó ba nghiên cứu (Bakal và c.s., 2025; Peng và c.s., 2023; Vaithilingam và c.s., 2022) sử dụng biến đo trực tiếp thời gian, và một nghiên cứu (Ziegler và c.s., 2024) sử dụng tỉ lệ người dùng chấp thuận làm proxy. Kết quả tổng hợp trung bình có trọng số khi bao gồm toàn bộ bốn nghiên cứu cho thấy việc áp dụng vibe coding giúp giảm thời gian phát triển trung bình 23,86%, với khoảng tin cậy 95% từ 23,56% đến 24,16 (Hình 2). Tuy nhiên, cần lưu ý rằng nghiên cứu của Ziegler et al. (Ziegler và c.s., 2024) sử dụng tỉ lệ chấp thuận thay cho thời gian thực tế và có cỡ mẫu rất lớn (n = 2.631), dẫn tới ảnh hưởng mạnh đến trọng số tổng hợp.

Khi loại bỏ Ziegler et al. để chỉ giữ lại các nghiên cứu đo trực tiếp, mức tiết kiệm thời gian trung bình đạt 19,04%

(95% CI: 18,56 - 19,51), như thể hiện trong Hình 2. Phân tích nhạy khẳng định kết quả này tương đối bền vững. Cụ thể, khi loại nghiên cứu có cỡ mẫu nhỏ và hiệu ứng tiêu cực là Vaithilingam et al. (2022), giá trị trung bình tăng nhẹ lên 20,34% (95% CI: 19,86 -20,83). Ngược lại, khi loại nghiên cứu thực nghiệm ngẫu nhiên duy nhất là Peng et al. (2023), mức tiết kiệm thời gian giảm nhẹ còn 18,70% (95% CI: 18,22 - 19,17).

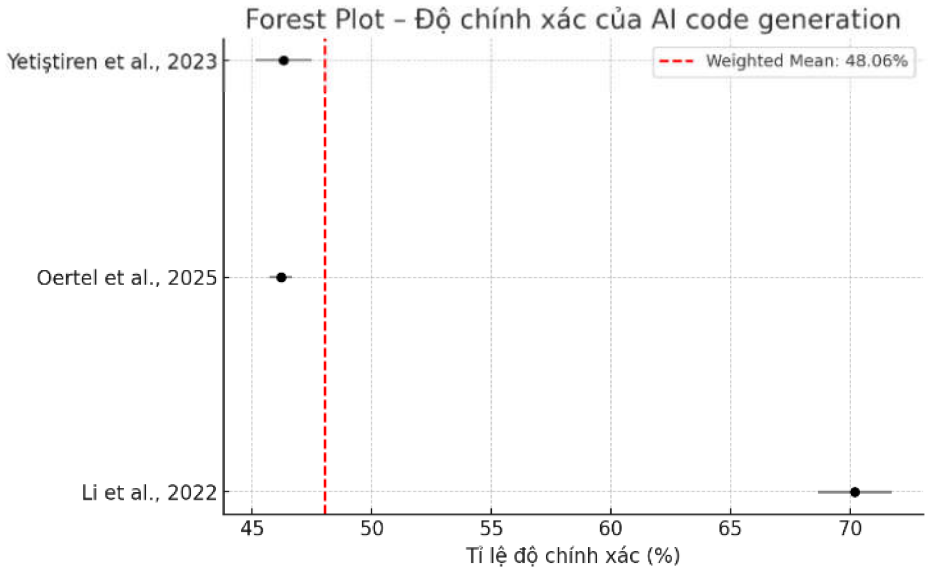
Kết quả này cho thấy tác động tích cực của vibe coding lên tốc độ phát triển phần mềm. Trong mọi kịch bản, giá trị trung bình dao động từ 18% đến 23%, với khoảng tin cậy hẹp và không có thay đổi đáng kể khi thực hiện phân tích nhạy.

Điều này phản ánh rằng ảnh hưởng của vibe coding tới tốc độ là hiện tượng có thật, có ý nghĩa thực tiễn và ổn định bất chấp sự khác biệt về bối cảnh và cỡ mẫu nghiên cứu.



Hình 2. Tác động của vibe coding lên tốc độ phát triển phần mềm – bao gồm và không bao gồm Ziegler et al., 2024.

3.2. Chất lượng phần mềm



Hình 3. Độ chính xác của mã sinh bởi AI

Ba nghiên cứu cung cấp dữ liệu định lượng liên quan đến độ chính xác của mã sinh bởi công cụ vibe coding bao gồm Li et al. (Li và c.s., 2022), Oertel et al. (Oertel và c.s., 2025), và Yetiştiren et al. (Yetiştiren và c.s., 2023). Kết quả tổng hợp trung bình có trọng số cho thấy độ chính xác của mã đạt mức 48,06%, với khoảng tin cậy 95% từ 47,63% đến 48,48% (Hình 3). Kết quả này phản ánh xác suất trung bình để một đoạn mã sinh bởi hệ thống AI đúng ngay từ lần đầu.

Tiếp đó, phân tích nhạy được thực hiện để kiểm tra độ bền vững của kết luận. Khi loại bỏ nghiên cứu của Oertel et al. (2025) - nghiên cứu có cỡ mẫu lớn nhất - giá trị trung bình tăng đáng kể lên 55,10% (95% CI: 54,17 - 56,03), cho thấy rằng nghiên cứu này có tác động làm giảm trung bình do độ chính xác dao động thấp trong các bài toán phức tạp hoặc không quen thuộc với mô hình. Ngược lại, khi loại nghiên cứu của Li et al. (2022), vốn có độ chính xác cao trên tập HumanEval (70,2%), mức trung bình giảm xuống còn 46,21% (95% CI: 45,77 - 46,66). Loại nghiên cứu của Yetiştiren et al. (2023) không tạo ra sự khác biệt đáng kể, với giá trị trung bình đạt 48,32% (95% CI: 47,87 - 48,78), gần tương đương với kết quả tổng hợp ban đầu.

Từ những kết quả trên, có thể nhận thấy rằng mặc dù tồn tại sự khác biệt đáng kể về độ chính xác giữa các bối cảnh bài toán, độ chính xác trung bình của các hệ thống vibe coding ổn định trong khoảng 46% đến 55%, với độ tin cậy cao. Đồng nghĩa rằng các công cụ AI hiện tại có thể tạo ra mã đúng ngay từ lần đầu trong khoảng một nửa số lần thử, phụ thuộc mạnh vào tính chất nhiệm vụ, phạm vi dữ liệu huấn luyện và mức độ lặp lại của các mẫu code.

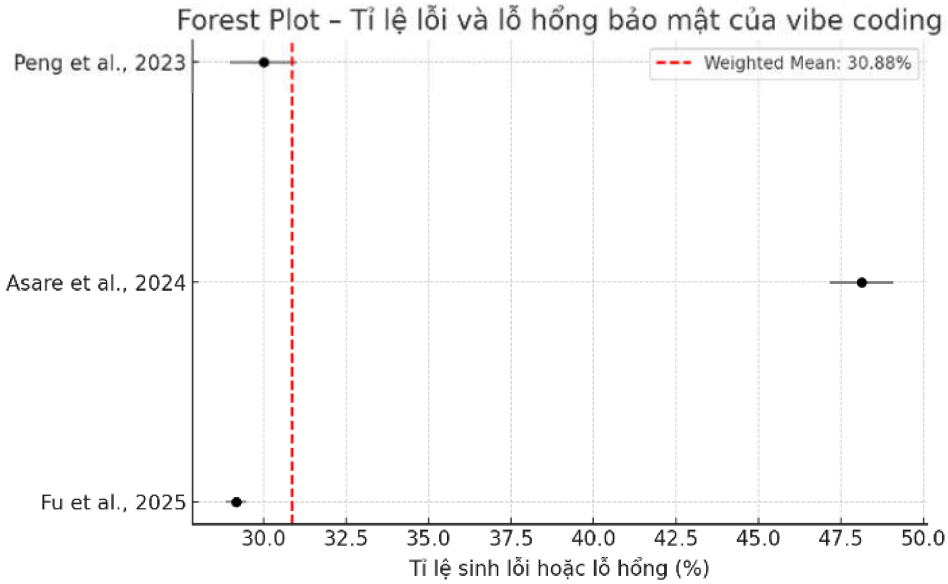
3.3. Bảo mật và nợ kỹ thuật

3.3.1. Tỉ lệ lỗi

Ba nghiên cứu cung cấp dữ liệu định lượng liên quan đến tỉ lệ lỗi và lỗ hổng bảo mật trong mã được sinh bởi các công cụ vibe coding, bao gồm Fu et al. (Fu và c.s., 2025), Asare et al. (Asare và c.s., 2024) và Peng et al. (Peng và c.s., 2023). Kết quả tổng hợp chỉ ra rằng trung bình có khoảng 30,88% mã sinh ra chứa lỗi hoặc lỗ hổng bảo mật, với khoảng tin cậy 95% nằm trong khoảng 30,60 - 31,17 (Hình 4). Con số này phản ánh nguy cơ lỗi hoặc bất toàn về bảo mật trong lần sinh mã đầu tiên khi sử dụng các công cụ AI hỗ trợ lập trình.

Tuy nhiên, mức độ rủi ro không

hoàn toàn đồng nhất giữa các nghiên cứu. Khi loại bỏ nghiên cứu của Fu et al. (2025) - nghiên cứu có cỡ mẫu lớn nhất và tỉ lệ lỗi thấp nhất (29,17%), giá trị trung bình tăng đáng kể lên 39,48% (95% CI: 38,79 - 40,18). Điều này chỉ ra rằng nghiên cứu của Fu đóng vai trò làm giảm trung bình chung nhờ cỡ mẫu lớn và quy trình đánh giá nghiêm ngặt bằng static scan. Ngược lại, loại bỏ Asare et al. (2024) - vốn báo cáo tỉ lệ lỗ hổng bảo mật cao hơn (48,14%) - khiến giá trị trung bình giảm xuống 29,24% (95% CI: 28,95 - 29,54), cho thấy nghiên cứu này có tác động làm lệch trung bình theo hướng cao hơn. Đáng chú ý, việc loại Peng et al. (2023) hầu như không làm thay đổi đáng kể kết quả, khi giá trị trung bình vẫn giữ ở mức 30,96% (95% CI: 30,66 - 31,25), điều này phản ánh rằng tỉ lệ lỗi trong nghiên cứu này gần với giá trị trung bình tổng.



Hình 4. Tỉ lệ lỗi hoặc lỗ hổng trong mã sinh bởi AI

Tổng thể, kết quả cho thấy rằng rủi ro xuất hiện lỗi hoặc lỗ hổng trong mã sinh bởi vibe coding dao động trong khoảng 29% đến 39%. Mặc dù con số này không đủ cao để phủ nhận hiệu quả của công cụ, nó cũng đủ để xác nhận rằng kiểm thử và rà soát thủ công vẫn là điều kiện bắt buộc trong quy trình phát triển phần mềm có sự hỗ trợ của AI.

3.3.2. Tỉ lệ sửa lỗi thành công

Dữ liệu định lượng từ hai nghiên cứu, Liu et al. (Liu và c.s., 2024) và Fu et al. (Fu và c.s., 2025), cho thấy khả năng sửa lỗi của hệ thống vibe coding chịu ảnh hưởng đáng kể bởi cách thức tương tác với mô hình. Trong nghiên cứu của Liu et al. (2024), khi áp dụng quy trình tương tác nhiều vòng (multi-round feedback) kết hợp với hướng dẫn cụ thể dựa trên mã lỗi và tiêu chuẩn bảo mật (CWE), tỉ lệ sửa lỗi đạt mức rất cao, vượt quá 89%.

Ngược lại, nghiên cứu của Fu et al. (2025) tập trung vào khả năng sửa lỗi trong một lượt tương tác duy nhất. Khi

chỉ sử dụng lệnh sửa mặc định (/fix), hệ thống chỉ khắc phục thành công 19,3% lỗ hổng. Tuy nhiên, khi prompt được bổ sung thông tin cụ thể về lỗi (bao gồm cảnh báo từ công cụ phân tích tĩnh hoặc mô tả lỗi), tỉ lệ sửa lỗi tăng đáng kể lên 55,5%. Điều này cho thấy rằng chất lượng của prompt và bối cảnh thông tin đóng vai trò then chốt trong việc nâng cao khả năng sửa lỗi.

Nhìn chung, kết quả cho thấy các công cụ vibe coding có thể đạt được hiệu suất sửa lỗi cao, đặc biệt khi quy trình tương tác được thiết kế hợp lý. Tuy nhiên, trong các tình huống thực tế khi sửa lỗi chỉ qua một lượt prompt, hiệu suất sửa lỗi còn hạn chế nếu thiếu thông tin hướng dẫn đầy đủ. Ngoài ra, nghiên cứu của Liu et al. (2024) cũng chỉ ra rằng độ phức tạp của mã có xu hướng tăng dần qua mỗi vòng sửa, làm dấy lên lo ngại về nợ kỹ thuật và khả năng bảo trì trong các quy trình sửa lỗi tự động dựa trên AI.

V. Thảo luận

Kết quả tổng hợp từ mười bảy nghiên cứu cho thấy rằng vibe coding mang lại những lợi ích rõ rệt về mặt tốc độ phát triển phần mềm, nhưng đồng thời cũng đặt ra các vấn đề nghiêm trọng liên quan đến độ chính xác, bảo mật và nợ kỹ thuật. Trung bình, tốc độ phát triển được cải thiện từ 19% đến 23%, phản ánh hiệu quả đáng kể trong việc giảm thời gian viết mã. Tuy nhiên, mức độ cải thiện này không đi kèm với sự gia tăng tương ứng về chất lượng. Độ chính xác trung bình của mã sinh ra chỉ đạt 48%, trong khi tỉ lệ lỗi hoặc lỗ hổng bảo mật trong lần sinh đầu tiên lên tới 31%.

Phân tích sâu hơn cho thấy hiệu suất của vibe coding phụ thuộc mạnh mẽ vào cách thức tương tác giữa lập trình viên và hệ thống AI. Cụ thể, khả năng sửa lỗi có thể tăng từ 19,3% lên tới 55,5% khi sử dụng prompt được thiết kế tốt, và thậm chí vượt quá 89% nếu áp dụng quy trình tương tác nhiều vòng (multi-round feedback) kết hợp với hướng dẫn từ cảnh báo bảo mật. Quan sát này nhấn mạnh rằng chất lượng của mã sinh bởi AI không phải là hằng số, mà phụ thuộc vào mức độ tham gia tích cực của con người trong quá trình hiệu chỉnh và kiểm soát.

Một vấn đề đáng lo ngại là sự gia tăng độ phức tạp của mã sau quá trình sửa lỗi qua nhiều vòng, như được ghi nhận trong nghiên cứu của Liu et al. (2024). Điều này tiềm ẩn nguy cơ tích lũy nợ kỹ thuật, làm suy giảm khả năng bảo trì và mở rộng của phần mềm trong dài hạn. Việc này đặc biệt nghiêm trọng trong bối cảnh mã sinh bởi AI ngày càng được tích hợp vào quy trình phát triển phần mềm thực tế.

Ngoài ra, kết quả cũng chỉ ra sự mất cân đối giữa tốc độ và chất lượng. Trong khi vibe coding có thể tăng tốc độ lập trình lên đáng kể, nó không đảm bảo giảm thiểu rủi ro liên quan đến lỗi hoặc lỗ hổng bảo mật nếu không đi kèm với quy trình kiểm thử và rà soát nghiêm ngặt. Điều kiện này làm nổi bật vai trò không thể thay thế của

con người trong quá trình phát triển phần mềm có sự hỗ trợ của AI.

Từ góc độ nghiên cứu, vẫn tồn tại một khoảng trống đáng kể trong việc đánh giá tác động dài hạn của vibe coding lên khả năng bảo trì (maintainability), tích lũy nợ kỹ thuật (technical debt) và độ tin cậy của phần mềm. Đặc biệt, các mô hình cộng tác giữa con người và AI trong môi trường phát triển phần mềm hiện tại mới chỉ ở giai đoạn sơ khai, cần được nghiên cứu sâu hơn để tối ưu hóa hiệu suất và giảm thiểu rủi ro.

V. Kết luận

Bài báo này đã hoàn thành ba mục tiêu nghiên cứu chính đề ra trong phần mở đầu. Thứ nhất, thông qua tổng hợp có hệ thống các bằng chứng thực nghiệm mới nhất, bài nghiên cứu xác nhận rằng vibe coding có tác động tích cực đến tốc độ phát triển phần mềm với mức cải thiện trung bình từ 19% đến 23%, nhưng đồng thời độ chính xác của mã sinh ra chỉ đạt 48%, và tỉ lệ lỗi hoặc lỗ hổng bảo mật trung bình đạt 31% trong lần sinh đầu tiên.

Thứ hai, bài nghiên cứu đã làm rõ các mối nguy tiềm ẩn liên quan đến bảo mật và nợ kỹ thuật. Mặc dù các công cụ vibe coding có khả năng sửa lỗi với hiệu suất cao, đặc biệt khi áp dụng quy trình tương tác nhiều vòng hoặc sử dụng prompt giàu ngữ cảnh, quá trình này đi kèm với sự gia tăng đáng kể về độ phức tạp của mã. Đây là chỉ dấu rõ ràng cho thấy sự tích lũy nợ kỹ thuật và nguy cơ suy giảm khả năng bảo trì phần mềm nếu không có biện pháp kiểm soát phù hợp.

Thứ ba, bài nghiên cứu đưa ra các định hướng rõ ràng cho nghiên cứu tương lai. Đặc biệt, cần có các nghiên cứu chuyên sâu về mô hình cộng tác con người–AI trong phát triển phần mềm, tập trung vào các yếu tố như tối ưu hóa chiến lược prompt, tự động hóa quy trình sửa lỗi mà vẫn kiểm soát được nợ kỹ thuật, và phát triển các công cụ hỗ trợ kiểm thử và rà soát mã sinh bởi AI.

Tổng thể, kết quả nghiên cứu khẳng định rằng vibe coding có thể trở thành một phần hữu ích trong quy trình phát triển phần mềm hiện đại, nhưng chỉ khi nó được sử dụng trong một khung kiểm soát chặt chẽ, với sự phối hợp hiệu quả giữa con người và AI. Việc lạm dụng hoặc thiếu kiểm soát có thể dẫn tới hệ quả ngược lại, làm gia tăng rủi ro bảo mật và chi phí kỹ thuật trong dài hạn.

Tài liệu tham khảo:

[1]. Andrej Karpathy [@karpathy]. (2025, Tháng Hai 2). *There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper* [Tweet]. Twitter. https://x.com/karpathy/status/1886192184808149383

[2]. Asare, O., Nagappan, M., & Asokan, N. (2024). *Is GitHub's Copilot as Bad as Humans at Introducing Vulnerabilities in Code?* (No. arXiv:2204.04741). arXiv. https://doi.org/10.48550/

arXiv.2204.04741

[3]. Bakal, G., Dasdan, A., Katz, Y., Kaufman, M., & Levin, G. (2025). *Experience with GitHub Copilot for Developer Productivity at Zoominfo* (No. arXiv:2501.13282). arXiv. https://doi.org/10.48550/arXiv.2501.13282

[4]. Barke, S., James, M. B., & Polikarpova, N. (2022). *Grounded Copilot: How Programmers Interact with Code-Generating Models* (No. arXiv:2206.15000). arXiv. https://doi.org/10.48550/arXiv.2206.15000

[5]. Bird, C., Ford, D., Zimmermann, T., Forsgren, N., Kalliamvakou, E., Lowdermilk, T., & Gazit, I. (2023). Taking Flight with Copilot. *Communications of the ACM*, *66*(6), 56–62. https://doi.org/10.1145/3589996

[6]. Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. *Proceedings of the 24th Australasian Computing Education Conference*, 10–19. https://doi.org/10.1145/3511861.3511863

[7]. Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2025). *Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study* (No. arXiv:2310.02059). arXiv. https://doi.org/10.48550/arXiv.2310.02059

[8]. Hong, Q. N., Pluye, P., Fàbregues, S., Bartlett, G., Boardman, F., Cargo, M., Dagenais, P., Gagnon, M.-P., Griffiths, F., Nicolau, B., O'Cathain, A., Rousseau, M.-C., & Vedel, I. (2019). Improving the content validity of the mixed methods appraisal tool: A modified e-Delphi study. *Journal of Clinical Epidemiology*, *111*, 49-59.e1. https://doi.org/10.1016/j.jclinepi.2019.03.008

[9]. Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A. D., Hubert, T., Choy, P., d'Autume, C. de M., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Cherepanov, A., … Vinyals, O. (2022). Competition-Level Code Generation with AlphaCode. *Science*, *378*(6624), 1092–1097. https://doi.org/10.1126/science.abq1158

[10]. Liang, J. T., Yang, C., & Myers, B. A. (2024). A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 1–13. https://doi.org/10.1145/3597503.3608128

[11]. Liu, Z., Tang, Y., Luo, X., Zhou, Y., & Zhang, L. F. (2024). *No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT* (No. arXiv:2308.04838). arXiv. https://doi.org/10.48550/arXiv.2308.04838

[12]. Oertel, J., Klünder, J., & Hebig, R. (2025). Don't settle for the first! How many GitHub Copilot solutions should you check? *Information and Software Technology*, *183*, 107737. https://doi.org/10.1016/j.infsof.2025.107737

[13]. Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J.

M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, *372*, n71. https://doi.org/10.1136/bmj.n71

[14]. Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot* (No. arXiv:2302.06590). arXiv. https://doi.org/10.48550/arXiv.2302.06590

[15]. Roose, K. (2025, Tháng Hai 27). Not a Coder? With A.I., Just Having an Idea Can Be Enough. *The New York Times*. https://www.nytimes.com/2025/02/27/technology/personaltech/vibecoding-ai-software-programming.html

[16]. Solohubov, I., Moroz, A., Tiahunova, M. Y., Kyrychek, H. H., & Skrupsky, S. (không ngày). *Accelerating software development with AI: exploring the impact of ChatGPT and GitHub Copilot*.

[17]. Tabachnyk, M., & Nikolov, S. (2022, Tháng Bảy 26). *ML-Enhanced Code Completion Improves Developer Productivity*. https://research.google/blog/ml-enhanced-code-completion-improves-developer-productivity/

[18]. Tyndall, J. (2010). *AACODS Checklist: Appraisal tool for grey literature* (tr 2) [Checklist]. Flinders University. https://fac.flinders.edu.au/dspace/api/core/bitstreams/e94a96eb-0334-4300-8880-c836d4d9a676/content

[19]. 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*, 1–7. https://doi.org/10.1145/3491101.3519665

[20]. Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). *Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT* (No. arXiv:2304.10778). arXiv. https://doi.org/10.48550/arXiv.2304.10778

[21]. Zhang, B., Liang, P., Zhou, X., Ahmad, A., & Waseem, M. (2023). *Practices and Challenges of Using GitHub Copilot: An Empirical Study*. 124–129. https://doi.org/10.18293/SEKE2023-077

[22]. Ziegler, A., Kalliamvakou, E., Li, X. A., Rice, A., Rifkin, D., Simister, S., Sittampalam, G., & Aftandilian, E. (2024). Measuring GitHub Copilot's Impact on Productivity. *Communications of the ACM*, *67*(3), 54–63. https://doi.org/10.1145/3633453

Phụ lục

Bảng trích xuất dữ liệu từ 17 nghiên cứu trong tổng quan hệ thống

ID	Tác giả – Năm	Loại tài liệu	Bối cảnh nghiên cứu	Cỡ mẫu / Quy mô	Công cụ AI sinh mã	Ngôn ngữ lập trình	Chỉ số tốc độ	Chỉ số chất lượng	Kết quả chính	Đánh giá chất lượng (MMAT/AACODS)
1	Liang et al., 2024	Bài báo hội nghị (ICSE)	Khảo sát lập trình viên quy mô lớn	410 lập trình viên	Copilot, Tabnine, CodeWhisperer, ChatGPT	Python, JavaScript, HTML/CSS, …	Trung vị (Md) = 30,5 % mã do AI hỗ trợ	Các vấn đề không đáp ứng yêu cầu NFR	Động lực là giảm gõ phím, nhanh hoàn thành; hạn chế ở kiểm soát chất lượng	Trung bình
2	Solohubov et al., 2024	CEUR W. Proceedings	Thí nghiệm nhỏ nhiều tác vụ	Không nêu rõ	ChatGPT, Copilot	Dart, Flutter, JS, …	≈ 30 % nhanh hơn viết logic	Không giảm chất lượng; ít lỗi	AI hỗ trợ học ngôn ngữ mới & CRUD nhanh	Thấp
3	Li et al., 2022	Science (AlphaCode)	Mô phỏng thi Codeforces	10 cuộc thi, 5k+ submissions	AlphaCode	C++, Python	Không đo; tập trung năng lực giải bài	Giải 34 % bài, top 54 % thi đấu	AlphaCode giải 34 % bài và xếp hạng ngang lập trình viên trung bình trên Codeforces, chứng tỏ AI có thể cạnh tranh ở cấp độ con người	Cao
4	Oertel et al., 2025	Tạp chí IST	Phân tích 17 048 gợi ý Copilot	2 025 bài toán, 17 048 gợi ý	GitHub Copilot	Nhiều (Python, JS, …)	Không đo; tập trung độ chính xác	Tăng xác suất đúng khi xem ≥ 5 gợi ý	Nên xem 4–5 gợi ý Copilot để nâng xác suất lời giải đúng; chỉ dùng gợi ý đầu thường thiếu chính xác	Cao
5	Yetiştiren et al., 2023	Bài báo journal (Empirical Study)	Benchmark HumanEval 164 bài	164 bài + 3 công cụ	Copilot, CodeWhisperer, ChatGPT	Python	Không đo	ChatGPT đúng 65,2 %; Copilot 46,3 %	ChatGPT đạt tỷ lệ giải đúng cao nhất; tuy nhiên Copilot và CodeWhisperer vẫn có ưu thế riêng tùy ngữ cảnh	Cao
6	Vaithilingam et al., 2022	Extended Abstract CHI	Thí nghiệm trong phòng, 24 lập trình viên	24 lập trình viên, within subjects	GitHub Copilot	Python	Không cải thiện thời gian hoàn thành	Tỉ lệ hoàn thành không đổi; khó chỉnh sửa gợi ý	19/24 thích dùng Copilot vì tạo điểm khởi đầu nhưng gặp khó khăn khi đọc hiểu và debug mã	Trung bình
7	Bakal et al., 2025	White paper doanh nghiệp	Triển khai thực tế tại Zoominfo	400+ lập trình viên	GitHub Copilot	TypeScript, Java, Python, JS	Ước tính tiết kiệm ~20 % thời gian	Tỉ lệ chấp nhận 33 % gợi ý; 20 % dòng code	Copilot nâng cao năng suất và mức độ hài lòng (72 %) nhưng gặp khó khăn trong việc hiểu logic chuyên biệt trong từng lĩnh vực, chất lượng mã không đồng đều, và người dùng cần cẩn thận với những vấn đề bảo mật	AACODS – Trung bình
8	Barke et al., 2022	OOPSLA	Grounded theory, 20 lập trình viên	20 lập trình viên, 4 ngôn ngữ	GitHub Copilot	Python, Rust, Haskell, Java	Không đo định lượng; phân loại chế độ tăng tốc & khám phá	Nêu khó khăn / chiến lược xác thực gợi ý	Xác định hai chế độ tương tác với Copilot: (1) tăng tốc – lập trình viên đã biết giải pháp và dùng Copilot để gõ nhanh; (2) khám phá – lập trình viên chưa rõ hướng và dùng Copilot để thử nghiệm ý tưởng; bài báo đề xuất thiết kế trợ lý phù hợp từng chế độ	Trung bình
9	Asare et al., 2024	Empirical Software Engineering	Phân tích bảo mật 152 mẫu C/C++	152 mẫu prompt Big Vul	GitHub Copilot	C/C++	Không đo	Lặp lại lỗ hổng 33 %; gợi ý bản vá 25 %	Copilot không tệ hơn con người trong nhiều trường hợp nhưng vẫn dễ tái sinh lỗ hổng cũ	Cao
10	Ziegler et al., 2024	Communications of the ACM	Khảo sát + telemetry 2 631 dev	2 631 lập trình viên	GitHub Copilot	Đa ngôn ngữ, chủ yếu Python, JS	Tỉ lệ chấp nhận gợi ý 27 %; DCPU > 312	Cảm nhận năng suất cao hơn khi có gợi ý hữu ích	Lợi ích năng suất phản ánh qua tỉ lệ chấp thuận; junior dev hưởng lợi nhiều nhất	Cao
11	Liu et al., 2024	Journal (preprint)	Đánh giá 728 bài & 54 tình huống CWE	728 bài lập trình, 18 CWE × 54 scenarios	ChatGPT (GPT 3.5)	C, C++, Java, Python, JavaScript	Không đo	Giải đúng 48,14 % bài toán phát hành trước 2021; sau nhiều vòng hội thoại, ChatGPT sửa được hơn 89 % lỗ hổng bảo mật	ChatGPT đạt độ chính xác trung bình; quy trình sửa lỗi qua nhiều vòng hội thoại phức tạp và chỉ khắc phục được khoảng 89 % lỗ hổng bảo mật	Cao
12	Zhang et al., 2023	Conference (SEKE)	Phân tích 169 SO + 655 discussions	824 bài thảo luận	GitHub Copilot	JS, Python chủ yếu	Không đo, người dùng nói giảm gõ phím	Lợi ích: mã có hoạt động; Hạn chế: khó tích hợp	Copilot như con dao hai lưỡi, tăng hiệu quả nhưng gây khó kiểm soát quy trình	Trung bình
13	Fu et al., 2025	ACM TOSEM (in press)	733 snippets Copilot trong dự án GitHub	733 đoạn mã	GitHub Copilot	Python, JavaScript	Không đo	29,5 % Python & 24,2 % JS có lỗ hổng; 43 CWE; 55,5 % fix qua Copilot Chat	Xác nhận nguy cơ bảo mật cao; Copilot Chat chỉ sửa được phân nửa	Cao
14	Bird et al., 2023	Communications of the ACM	Phân tích forum + case study + survey 2047 dev	2047 khảo sát + 279 posts + 5 dev	GitHub Copilot	Đa ngôn ngữ	Người dùng cảm giác năng suất tăng; bớt tìm StackOverflow	Tăng thời gian review, acceptance rate tương quan productivity	Lập trình cùng AI làm thay đổi bộ kỹ năng của lập trình viên; họ dành nhiều thời gian đọc và đánh giá mã do AI sinh hơn là tự viết	Trung bình
15	Peng et al., 2023	Controlled trial (arXiv)	RCT 95 dev task HTTP server	95 dev (45 treatment, 50 control)	GitHub Copilot	JavaScript	Nhóm Copilot hoàn thành nhanh hơn 55,8 % (71 vs 161 phút)	Tỉ lệ hoàn thành: 78 % vs 70 %	Copilot tăng tốc đáng kể, lợi ích mạnh hơn với dev ít kinh nghiệm	Cao
16	Finnie Ansley et al., 2022	Conference (ACE)	So sánh Codex với sinh viên CS1	23 câu hỏi thi + 7 biến thể Rainfall; 50 mẫu/biến	OpenAI Codex	Python chủ yếu	Không đo	Codex đạt ~78 % điểm, xếp top 25 % lớp; giải 70 % bài HumanEval	Codex vượt đa số sinh viên nhập môn; làm dấy lên thách thức về liêm chính học thuật và thiết kế đánh giá	Trung bình
17	Tabachnyk & Nikolov, 2022	Google AI Blog (white paper)	Triển khai nội bộ Google	10 000+ lập trình viên, 3 tháng	ML Enhanced Code Completion (Transformer)	C++, Java, Go, Python	Thời gian lập build–test giảm ~6 %	≈ 3 % dòng code do AI; không làm gián đoạn quy trình	Gợi ý ML tăng nhẹ năng suất thực tế mà không ảnh hưởng luồng công việc	AACODS – Trung bình

Phụ lục 1. Kết quả trích xuất dữ liệu từ các nghiên cứu liên quan đến vibe coding

VIBE CODING IN SOFTWARE DEVELOPMENT: A SYSTEMATIC REVIEW OF ITS IMPACT ON SPEED, QUALITY, AND TECHNICAL DEBT

Quach Thi Hanh[2], Nguyen To Uyen[2], Vu Tat Diep[2]

Abstract: *The rapid advancement of large language models (LLMs) has facilitated the emergence of "vibe coding"- a programming approach that enables AI to generate code based on natural language descriptions of requirements. This study conducts a systematic literature review (SLR) to synthesize the latest empirical evidence on the impact of vibe coding on software development speed and quality. Following the PRISMA 2020 guidelines, data were collected from 17 studies published between 2022 and 2025. The findings indicate that vibe coding improves development speed by an average of 19% to 23%. However, the accuracy of AI-generated code remains at 48%, and the defect or vulnerability rate for first-attempt code generation is approximately 31%, indicating significant quality risks. Nevertheless, the bug fix rate can reach up to 89% when applying multi-round feedback or prompts enriched with contextual information. This study clarifies the relationship between speed, quality, and security in the context of vibe coding and highlights the urgent need for developing more effective human - AI collaboration models to mitigate technical debt and ensure software reliability.*

Keywords: *vibe coding, large language model, Github Copilot, developer productivity, software quality*

[2] Hanoi Open University