PERANCANGAN FRAMEWORK VIBE CODING YANG BERTANGGUNG JAWAB DALAM PENGEMBANGAN PERANGKAT LUNAK

Eko Harli[1], Ahmad Fauzi[2], Tria Hadi Kusmanto[3]
[1,2,3]Teknik Informatika, Universitas Indraprasta PGRI
ekoharli@gmail.com[1], ahmadfauzi.udzi@gmail.com[2], triahadi226@gmail.com[3]

Submitted Februarry 27, 2026; Revised April 2, 2026; Accepted April 4, 2026

Abstrak

Perkembangan *large language models* (LLM) telah melahirkan fenomena *vibe coding*, paradigma pemrograman berbasis prompt yang meningkatkan produktivitas namun memunculkan risiko serius berupa penurunan pemahaman kode, kualitas *maintainability* rendah, akumulasi *technical debt*, dan kerentanan keamanan aplikasi. Penelitian ini bertujuan merancang *Framework Vibe Coding* yang Bertanggung Jawab yang mengintegrasikan prinsip *software engineering* (SOLID), *computational thinking*, kolaborasi manusia-AI, serta praktik pengembangan perangkat lunak yang aman dan mudah dipelihara. Menggunakan pendekatan kualitatif-konseptual melalui studi literatur sistematis dari publikasi 2015-2025, penelitian ini mengidentifikasi fenomena, menganalisis teori pendukung, dan mensintesis *framework* dengan lima komponen utama: *Input Layer* (spesifikasi intent), AI *Assistance Layer* (generasi kode LLM), *Human Oversight Layer* (verifikasi manusia), *Engineering Guardrails Layer* (pengujian otomatis dan *security scanning*), serta *Output Layer* (kode siap produksi). *Framework* ini memastikan pengambilan keputusan berpusat pada manusia, disiplin rekayasa perangkat lunak terintegrasi, serta transparansi dan akuntabilitas, sehingga mengatasi tantangan *vibe coding* secara komprehensif sambil mempertahankan kualitas, keamanan, dan pemahaman kode dalam konteks pendidikan dan industri.

Kata Kunci : *vibe coding*, kecerdasan buatan, rekayasa perangkat lunak, keamanan kode

Abstract

The advancement of large language models (LLM) has given rise to vibe coding, a prompt-based programming paradigm that enhances productivity but introduces serious risks including reduced code comprehension, low maintainability, accumulation of technical debt, and application security vulnerabilities. This research aims to design a Responsible Vibe Coding Framework that integrates software engineering principles (SOLID), computational thinking, human-AI collaboration, and secure, maintainable software development practices. Employing a qualitative-conceptual approach through systematic literature review of publications from 2015-2025, the study identifies the phenomenon, analyzes supporting theories, and synthesizes a framework comprising five core components: Input Layer (intent specification), AI Assistance Layer (LLM code generation), Human Oversight Layer (human verification), Engineering Guardrails Layer (automated testing and security scanning), and Output Layer (production-ready code). The framework ensures human-centered decision-making, integrated software engineering discipline, and transparency with accountability, comprehensively addressing vibe coding challenges while maintaining code quality, security, and comprehension in educational and industrial contexts.

Keywords : *vibe coding, AI, software engineering, code security*

1. PENDAHULUAN

Kemajuan pesat dalam kecerdasan buatan generatif, khususnya pada *large language models* (LLM), telah mengubah *landscape* pengembangan perangkat lunak secara fundamental. Alat-alat seperti GitHub Copilot, ChatGPT, dan Claude memungkinkan pengembang untuk

menghasilkan kode melalui interaksi berbasis percakapan dengan sistem AI, menandai era baru dalam praktik pemrograman yang disebut *vibe coding* [1]. Fenomena ini merepresentasikan paradigma pemrograman yang berpusat pada AI, di mana pengembang mengekspresikan maksud fungsional tingkat tinggi dan deskripsi kualitatif (*vibe*) yang diinginkan, sementara agen AI mentransformasi spesifikasi tersebut menjadi kode yang dapat dieksekusi [2].

Vibe coding menawarkan potensi peningkatan produktivitas dan demokratisasi pengembangan perangkat lunak [3]. Namun praktik ini juga menampilkan sejumlah tantangan serius. Pertama, penurunan pemahaman kode menjadi risiko signifikan ketika pengembang bergantung sepenuhnya pada sistem AI tanpa memverifikasi atau memahami logika kode yang dihasilkan. Kedua, kualitas dan *maintainability* kode yang dihasilkan sering kali tidak memenuhi standar industri, dengan temuan menunjukkan bahwa kode yang dihasilkan AI memiliki tingkat kerentanan keamanan yang lebih tinggi dibandingkan kode yang ditulis secara manual [4], [5]. Ketiga, akumulasi *technical debt* terjadi ketika kode yang tidak optimal secara berkelanjutan diintegrasikan ke dalam basis kode, meningkatkan biaya pemeliharaan jangka panjang. Keempat, keamanan aplikasi menjadi perhatian kritis karena LLM sering menghasilkan kode yang rentan terhadap serangan injeksi, kontrol akses yang lemah, dan validasi input yang tidak memadai.

Penelitian empiris menunjukkan bahwa kode yang dihasilkan oleh LLM secara konsisten mengandung kerentanan sesuai dengan *Common Weakness Enumeration* (CWE), termasuk *buffer overflow*, SQL *injection*, dan penggunaan fungsi yang tidak aman [6]. Meskipun ada upaya berkelanjutan untuk meningkatkan keamanan melalui *prompt engineering* dan *fine-tuning*, belum ada *framework* formal yang mengatur praktik *vibe coding* secara bertanggung jawab dalam konteks *software engineering*. Kesenjangan penelitian ini menjadi dasar urgensi untuk mengembangkan *framework* komprehensif yang mengintegrasikan prinsip-prinsip rekayasa perangkat lunak dengan kemampuan AI generatif, memastikan bahwa kolaborasi manusia-AI menghasilkan perangkat lunak yang berkualitas, aman, dan mudah dipelihara.

Prinsip-prinsip rekayasa perangkat lunak telah menjadi fondasi utama untuk menghasilkan perangkat lunak berkualitas tinggi selama beberapa dekade, dengan prinsip SOLID (*Single Responsibility*, *Open*/*Closed*, *Liskov Substitution*, *Interface Segregation*, *Dependency Inversion*) [7] .

Prinsip ini menekankan desain yang dapat dipelihara, fleksibel, dan dapat dikembangkan, di mana penelitian empiris menunjukkan penerapannya secara signifikan meningkatkan pemahaman kode serta mengurangi *technical debt* bahkan dalam konteks *machine learning* [8], sehingga dalam *vibe coding* prinsip-prinsip ini harus diintegrasikan melalui mekanisme verifikasi otomatis dan panduan pengembang agar kode hasil AI mematuhi standar desain yang ditetapkan. Sementara itu, *computational thinking* (CT) merupakan kemampuan menyelesaikan masalah, merancang sistem, dan memahami perilaku manusia melalui konsep ilmu komputer yang mencakup abstraksi, dekomposisi, pengenalan pola, serta desain algoritma [9]. Berfungsi sebagai jembatan kritis antara maksud manusia (*intent*) dan realisasi teknis AI dalam *vibe coding*, di mana pengembang wajib mempertahankan keterampilan CT untuk mengevaluasi, mengidentifikasi ketidaksesuaian, dan menyesuaikan kode AI secara logis, dengan *framework* bertanggung jawab yang mendorong pemeliharaan keterampilan ini melalui pembelajaran interaktif dan refleksi

kritis. Kolaborasi manusia-AI yang efektif melampaui pandangan AI sebagai sekadar alat melalui penekanan pada *bidirectional learning*, manajemen kepercayaan, serta diferensiasi peran antara agen manusia dan AI [10], [11], di mana *framework* seperti *Human-AI Collaboration and Adaptation Framework* (HACAF) menuntut kompatibilitas dengan *workflow existing*, dukungan organisasi, dan mekanisme *feedback* dinamis, sehingga dalam *vibe coding* kolaborasi ini difasilitasi oleh antarmuka intuitif, verifikasi transparan, serta kejelasan tanggung jawab masing-masing pihak. Akhirnya, keamanan dan *maintainability* sebagai atribut kualitas kritis dalam pengembangan perangkat lunak menunjukkan bahwa kode hasil LLM memiliki tingkat kerentanan CWE yang signifikan—bahkan 40-60% lebih tinggi dibandingkan *baseline [12]*.

Penelitian ini bertujuan untuk merancang *Framework* Vibe Coding yang Bertanggung Jawab yang mengintegrasikan prinsip-prinsip *software engineering*, *computational thinking*, serta kolaborasi manusia-AI untuk memandu praktik *vibe coding* yang etis dan berkualitas tinggi; mengintegrasikan peran manusia dan AI dalam pengembangan perangkat lunak dengan mendefinisikan tanggung jawab yang jelas, mekanisme *oversight*, serta *feedback loop* yang memastikan manusia tetap sebagai pengambil keputusan utama; serta menjaga kualitas, keamanan, dan pemahaman kode melalui *engineering guardrails* yang komprehensif termasuk pengujian otomatis, *security scanning*, dan mekanisme *code review*. Secara teoretis, penelitian ini mengisi kesenjangan dengan menyediakan *framework* formal yang menghubungkan praktik *vibe coding* dengan teori rekayasa perangkat lunak dan interaksi manusia-AI, berkontribusi pada pemahaman epistemologis paradigma pemrograman berbasis AI, serta menjadi fondasi untuk penelitian empiris lanjutan. Secara praktis, *framework* ini memberikan panduan terstruktur bagi mahasiswa untuk belajar *vibe coding* dengan standar kualitas yang jelas, membantu pengembang profesional meningkatkan produktivitas sambil mempertahankan kontrol keamanan, menyediakan kurikulum bagi pendidik dalam pengajaran pemrograman era AI, serta menawarkan strategi adopsi bertanggung jawab bagi industri dengan mitigasi risiko yang terukur.

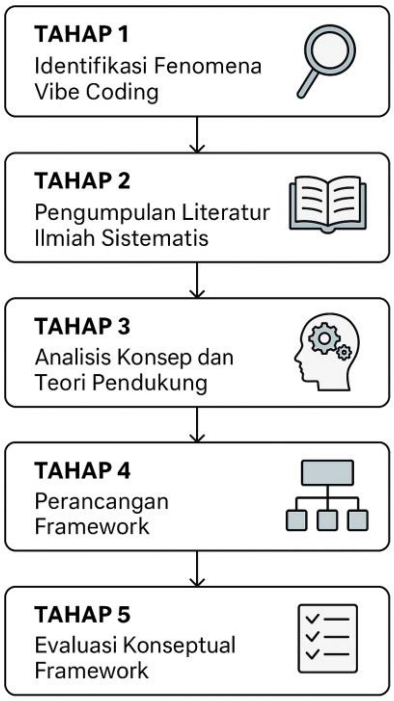
2. METODE PENELITIAN

Desain Penelitian

Penelitian ini menggunakan pendekatan kualitatif-konseptual dengan fokus pada perancangan *framework*. Metode ini dipilih karena memungkinkan eksplorasi mendalam terhadap fenomena *vibe coding* melalui analisis literatur sistematis dan interpretasi konseptual, tanpa bergantung pada data numerik. Selain itu, pendekatan ini mengintegrasikan temuan dari berbagai aliran penelitian, seperti *software engineering*, AI *safety*, *human-computer interaction*, dan pendidikan, untuk membangun landasan empiris yang kuat. *Framework* yang dihasilkan dirancang sebagai model konseptual komprehensif yang menghubungkan komponen kunci, proses interaksi, serta prinsip-prinsip pengaturan praktik *vibe coding* yang bertanggung jawab. Model ini divisualisasikan melalui deskripsi naratif dan diagramatik guna memfasilitasi pemahaman serta implementasi yang efektif.

Rancangan Kegiatan

Penelitian ini dilaksanakan melalui tahapan berikut:



Gambar 1. Tahapan Penelitian

1. Identifikasi Fenomena *Vibe Coding*
   Tahap ini bertujuan mendefinisikan batasan *vibe coding* sebagai transisi dari instruksi deterministik (pemrograman yang kaku) ke interpretasi probabilistik (pemrograman yang lebih fleksibel). Identifikasi difokuskan pada perubahan peran pengembang dari pelaksana teknis menjadi instruktur orkestrasi AI. Secara teoretis, hal ini dipahami sebagai konfigurasi ulang mediasi intensi yang mengubah struktur kognitif kerja profesional pengembang [13].
2. Pengumpulan Literatur Ilmiah Sistematis
   Tahap ini menggunakan metode *Systematic Literature Review* (SLR) untuk mengumpulkan data sekunder dari bidang *Software Engineering*, dan *AI Safety*. SLR memberikan sintesis mengenai kondisi pengetahuan dalam suatu bidang, yang dapat digunakan untuk mengidentifikasi prioritas penelitian [14].
3. Analisis Konsep dan Teori Pendukung
   Tahap ini berfokus pada analisis kualitatif untuk mengekstraksi konsep dan hubungan antar-variabel. Proses ini menyintesis berbagai teori guna membangun fondasi konseptual yang kokoh sebelum perancangan model. Pendekatan sistematis tersebut krusial untuk mengintegrasikan teori ke dalam kerangka evaluasi dan inkuiri yang komprehensif [15].
4. Perancangan *Framework*
   Tahap ini berfokus pada perancangan model konseptual yang mengintegrasikan komponen utama, proses interaksi, dan prinsip pengaturan. *Framework* ini berfungsi sebagai panduan praktis untuk implementasi *vibe coding* yang bertanggung jawab. Pengembangan struktur model tersebut dilakukan secara sistematis dan replikabel untuk memastikan ketajaman fondasi konseptualnya [16].
5. Evaluasi Konseptual *Framework*
   Tahap akhir berfokus pada evaluasi konseptual untuk menguji logika internal dan konsistensi teoretis *framework*. Pada tahap ini, pengujian terbatas pada validitas konsep guna memperkuat struktur model sebelum dilakukan implementasi praktis di masa depan.

Teknik Pengumpulan dan Analisis Data

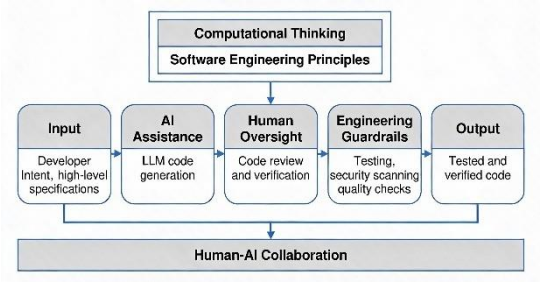
Sumber Data dari penelitian ini merupakan hasil Literatur ilmiah dari Jurnal bereputasi dan Prosiding konferensi terkemuka, serta buku akademik dan *handbook* tentang *software engineering*, AI *safety*, dan interaksi manusia dengan komputer.

Analisis data dilakukan melalui analisis tematik untuk mengidentifikasi dan mengorganisir tema kunci dari literatur mencakup karakteristik *vibe coding*, tantangan teknis-sosial, *best practices software engineering*, serta prinsip responsible AI; analisis komparatif untuk membandingkan temuan studi guna mengidentifikasi konvergensi-divergensi, *trade-offs* produktivitas versus kualitas/keamanan, dan pengaruh konteks-

spesifik; serta sintesis konseptual yang mengintegrasikan insights teori dan empiris untuk mengonstruksi *framework koheren*, menghubungkan komponen dengan fondasi teoritis, dan mengidentifikasi mekanisme yang menghubungkan input ke *outcome* yang diinginkan.

3. HASIL DAN PEMBAHASAN

Hasil penelitian menghasilkan sebuah *Framework Vibe Coding* yang Bertanggung Jawab yang dirancang untuk mengintegrasikan praktik *vibe coding* dengan prinsip-prinsip *software engineering* yang *established* dan *responsible AI principles*. Gambar 2 menyajikan rancangan Diagram Konseptual *Framework Vibe Coding*



Gambar 2. Diagram Konseptual Framework

Framework ini terdiri dari lima komponen utama yang berinteraksi dalam sebuah *iterative lifecycle*.

Komponen *Framework*

1. *Input Layer - Developer Intent Specification*
   Komponen ini merepresentasikan fase awal di mana pengembang mengartikulasikan maksud dan spesifikasi perangkat lunak yang akan dibangun. Input mencakup: *Functional Requirements*, *Qualitative Descriptors* (Vibe) dan *Contextual Information*.
   Tahap ini kritis karena kualitas input secara langsung mempengaruhi relevansi dan kualitas output AI. *Framework* menekankan pentingnya *computational thinking* dalam *formulating requirements*, mendorong pengembang untuk *break down complex problems*, *identify patterns*, dan berpikir logis tentang solusi desain yang diharapkan.
2. *AI Assistance Layer - Code Generation and Iteration*
   Komponen ini memanfaatkan kemampuan LLM untuk mengubah input menjadi *code artifacts*. Proses ini melibatkan *Prompt Engineering*, *Code Generation*, dan *Interactive Refinement*. Penelitian menunjukkan bahwa pengembang dengan latar belakang pemrograman tingkat lanjut menghasilkan *prompt* yang lebih kaya konteks dan mencapai hasil yang lebih baik dibandingkan pemula . *Framework* ini mengakomodasi heterogenitas tersebut dengan menyediakan *scaffolding* dan contoh yang membantu pengembang dari semua tingkat kemampuan.
3. *Human Oversight Layer - Code Review dan Verification*
   Komponen ini merupakan mekanisme kontrol yang memastikan manusia tetap menjadi pengambil keputusan utama. Peran manusia antara lain: *Functional Verification*, *Code Understanding*, *Quality Assessment*, dan *Security Review*.
4. *Engineering Guardrails Layer - Automated Quality Assurance*
   Komponen ini mengimplementasikan pemeriksaan dan pengimbangan sistematis untuk memastikan kualitas kode, keamanan, serta *maintainability*.
5. *Output Layer - Verified dan Production-Ready Code*
   Komponen ini merepresentasikan hasil akhir setelah melalui semua layer dan pengetesan. Output yang valid untuk hal ini harus lolos semua unit test, dan *security scan*.

Framework ini dibangun atas 3 prinsip fundamental, yaitu:

Prinsip 1: Pengambilan Keputusan Berpusat pada Manusia
Meskipun AI mampu menghasilkan kandidat kode, manusia tetap harus berperan sebagai pengambil keputusan utama pada setiap tahapan siklus pengembangan perangkat lunak. Framework yang diusulkan menjamin prinsip ini melalui penerapan lapisan *human oversight* yang bersifat wajib serta pembagian tanggung jawab yang jelas pada setiap tahap pengembangan.

Prinsip 2: Integrasi Disiplin Rekayasa Perangkat Lunak
Framework ini mengintegrasikan praktik-praktik rekayasa perangkat lunak yang telah mapan, seperti prinsip perancangan, pengujian, dan keamanan, ke dalam alur kerja *vibe coding*. Pendekatan ini tidak dimaksudkan untuk menggantikan disiplin rekayasa perangkat lunak, melainkan sebagai bentuk evolusi yang tetap mempertahankan ketelitian (*rigor*) dengan memanfaatkan AI sebagai sarana otomasi.

Prinsip 3: Transparansi dan Akuntabilitas
Seluruh komponen dalam *framework* dirancang dengan penekanan pada transparansi terhadap kemampuan dan keterbatasan AI, serta kejelasan akuntabilitas atas setiap keputusan yang diambil pada masing-masing tahapan. Pendekatan ini mendukung terbentuknya kepercayaan (*trust*) dan pengambilan keputusan yang berbasis informasi (*informed decision making*).

Pembahasan

Framework yang dirancang secara langsung mengatasi setiap tantangan signifikan yang teridentifikasi dalam latar belakang penelitian.

Mitigasi Penurunan Pemahaman Kode

Human Oversight Layer dengan penerapan persyaratan wajib pemahaman tentang kode pemrograman, memastikan bahwa pengembang tidak hanya menerima hasil keluaran AI secara pasif, tetapi terlibat secara aktif dalam proses analisis dan pengujian kode. Keterlibatan aktif ini mendorong terjadinya retensi pengetahuan serta pengembangan kemampuan *computational thinking* yang esensial dalam proses pemrograman.

Peningkatan Kualitas dan Maintainability. *Engineering Guardrails Layer* yang dilengkapi dengan pemeriksaan kualitas otomatis (*automated quality checks*) serta validasi prinsip perancangan perangkat lunak memastikan bahwa kode yang dihasilkan memenuhi standar kualitas yang telah ditetapkan. Dalam konteks *maintainability*, penerapan pemeriksaan kepatuhan terhadap prinsip SOLID dan kewajiban pembuatan dokumentasi (*mandatory documentation generation*) berperan signifikan dalam meningkatkan keterbacaan dan kemudahan pemeliharaan kode.

Reduksi *Technical Debt*

Melalui deteksi dini terhadap permasalahan desain menggunakan *static analysis* dan penerapan proses *human review* yang bersifat wajib, *framework* yang diusulkan mampu mencegah akumulasi kode berkualitas rendah yang secara tradisional menjadi penyumbang utama *technical debt*. Selain itu, mekanisme *iterative refinement loop* memungkinkan perbaikan dan penyempurnaan kode dilakukan sebelum tahap integrasi, sehingga potensi masalah dapat diminimalkan sejak awal.

Peningkatan Keamanan.

Pendekatan keamanan berlapis (*multi-layered security approach*) dalam *framework* ini mengombinasikan *security-aware prompting*, peninjauan keamanan oleh manusia, serta pemindaian kerentanan secara otomatis (*automated vulnerability scanning*). Kombinasi pendekatan tersebut terbukti efektif dalam mengurangi kerentanan pada kode yang dihasilkan oleh AI.

4. SIMPULAN

Penelitian ini berhasil merancang *Framework Vibe Coding* yang Bertanggung Jawab sebagai respons terhadap tantangan fenomena *vibe coding* dalam pengembangan perangkat lunak, yang mengintegrasikan prinsip-prinsip *software engineering* (SOLID), *computational thinking*, kolaborasi manusia-AI. *Framework* dengan lima komponen utama yaitu *Input Layer*, AI *Assistance Layer*, *Human Oversight Layer*, *Engineering Guardrails Layer*, serta *Output Layer*. *Framework* ini memastikan pengambilan keputusan berpusat pada manusia, disiplin rekayasa perangkat lunak terintegrasi, dan transparansi-akuntabilitas, sehingga secara komprehensif menjawab tujuan penelitian dengan mempertahankan kualitas, keamanan, serta pemahaman kode dalam konteks pendidikan dan industri.

Untuk penelitian lanjutan, disarankan melakukan validasi empiris *framework* melalui metode kuasi-eksperimen yang membandingkan produktivitas, kualitas kode, dan tingkat kerentanan antara tim yang menggunakan *framework* dengan praktik *vibe coding* tanpa *framework*; serta mengeksplorasi adaptasi *framework* dalam domain-spesifik seperti pengembangan aplikasi *web*, aplikasi *mobile* guna mengidentifikasi penyesuaian kontekstual yang diperlukan. Selain itu, penelitian pedagogis direkomendasikan untuk mengintegrasikan *framework* ke dalam kurikulum pemrograman tingkat sarjana dengan evaluasi *pre-post test* terhadap keterampilan *computational thinking* mahasiswa.

DAFTAR PUSTAKA

[1] I. Mehta, “A Quarter of Startups in YC’s Current Cohort Have Codebases That Are Almost Entirely AI-Generated.” Accessed: May 12, 2025. [Online]. Available: https://techcrunch.com/2025/03/06/a-quarter-of-startups-in-ycs-current-cohort-have-codebases-that-are-almost-entirely-ai-generated/

[2] M. Marron, “Toward Programming Languages for Reasoning: Humans, Symbolic Systems, and AI Agents,” in *Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, New York, NY, USA: ACM, Oct. 2023, pp. 136–152. doi: 10.1145/3622758.3622895.

[3] T. Weber, M. Brandmaier, A. Schmidt, and S. Mayer, “Significant Productivity Gains through Programming with Large Language Models,” *Proc. ACM Hum. Comput. Interact.*, vol. 8, no. EICS, pp. 1–29, Jun. 2024, doi: 10.1145/3661145.

[4] P. Sankhe, N. Patil, M. Ghorpade, P. Prasad, and M. Linkesh, “Empirical Analysis of AI-Assisted Code Generation Tools Impact on Code Quality, Security and Developer Productivity,” *International Journal For Multidisciplinary Research*, vol. 7, no. 6, Nov. 2025, doi: 10.36948/ijfmr.2025.v07i06.61350.

[5] T. Espinha Gasiba, A.-C. Iosif, I. Kessba, S. Amburi, U. Lechner, and M. Pinto-Albuquerque, “May the Source Be with You: On ChatGPT, Cybersecurity, and Secure Coding,” *Information*, vol. 15, no. 9, p. 572, Sep. 2024, doi: 10.3390/info15090572.

[6] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, “Asleep at the Keyboard? Assessing the Security of GitHub Copilot’s Code Contributions,” in *2022 IEEE Symposium on Security and Privacy (SP)*, IEEE, May 2022, pp. 754–768. doi: 10.1109/SP46214.2022.9833571.

[7] R. C. Martin, *Clean Code: A Handbook of Agile Software*

Craftsmanship, 2nd ed. Addison-Wesley Professional, 2025.

[8] R. Cabral, M. Kalinowski, M. T. Baldassarre, H. Villamizar, T. Escovedo, and H. Lopes, "Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, New York, NY, USA: ACM, Apr. 2024, pp. 7–17. doi: 10.1145/3644815.3644957.

[9] M. E. Rosadi, W. Wagino, N. Alamsyah, M. Rasyidan, and M. Y. Kurniawan, "Sosialisasi Computational Thinking untuk Guru-Guru di SDN Teluk dalam 3 Banjarmasin," *Jurnal SOLMA*, vol. 9, no. 1, pp. 45–54, Apr. 2020, doi: 10.29405/solma.v9i1.3352.

[10] V. Stray, A. Barbala, and V. T. Wivestad, "Human-AI Collaboration in Software Development: A Mixed-Methods Study of Developers' Use of GitHub Copilot and ChatGPT," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, New York, NY, USA: ACM, Jun. 2025, pp. 1325–1332. doi: 10.1145/3696630.3730566.

[11] S. Amershi *et al.*, "Guidelines for Human-AI Interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM, May 2019, pp. 1–13. doi: 10.1145/3290605.3300233.

[12] J. Peng, L. Cui, K. Huang, J. Yang, and B. Ray, "CWEval: Outcome-driven Evaluation on Functionality and Security of LLM Code Generation," in *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, IEEE, May 2025, pp. 33–40. doi: 10.1109/LLM4Code66737.2025.00009.

[13] C. Meske, T. Hermanns, E. Von der Weiden, K.-U. Loser, and T. Berger, "Vibe Coding as a Reconfiguration of Intent Mediation in Software Development: Definition, Implications, and Research Agenda," *IEEE Access*, vol. 13, pp. 213242–213259, 2025, doi: 10.1109/ACCESS.2025.3645466.

[14] M. J. Page *et al.*, "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews," *BMJ*, p. n71, Mar. 2021, doi: 10.1136/bmj.n71.

[15] M. Patton, *Qualitative Research and Evaluation Methods*, 4th ed. Thousand Oaks: Sage Publications, 2015.

[16] N. Tusquellas, D. López-Villanueva, R. Palau, and R. Santiago, "Educational Conceptual Model Design Research Methodology," *Universitas Tarraconensis Revista de Ciències de l Educació*, no. 2, p. e4103, Feb. 2025, doi: 10.17345/ute.2025.4103.