KAMPUS AKADEMIK PUBLISING
Jurnal Ilmiah Research Student (JIRS)
Vol.3, No.2 September 2026
e-ISSN: 3025-5694, p-ISSN : 3025-5708, Hal 41-59
DOI : https://doi.org/10.61722/jirs.v3i2.10626

Analysis of Technical Competency Degradation Risks in Developers Due to AI Code Assistant Usage: A Systematic Literature Review

Dandy Trisniprya Mardiecka[1], Evy Nurmiati[2]

[1,2] Fakultas Sains dan Teknologi, UIN Syarif Hidayatullah Jakarta
Jl. Ir. H. Djuanda No. 95, Ciputat, Kota Tangerang Selatan, Banten, Indonesia, 15412.
*Penulis Korespondensi: dandy.trisniprya24@mhs.uinjkt.ac.id

Abstract. *The widespread adoption of Artificial Intelligence (AI) code assistants such as GitHub Copilot, ChatGPT, and Amazon CodeWhisperer has significantly transformed software development workflows. While these tools offer documented productivity improvements, including up to 55% faster task completion, their uncritical adoption introduces serious risks to developer technical competency that remain insufficiently examined in the literature. This study aims to analyze the risks of technical competency degradation among software developers resulting from AI code assistant usage through a systematic literature review approach. Literature was collected from ScienceDirect, Scopus, and Google Scholar within the 2020–2026 period, selected through predefined inclusion and exclusion criteria, and analyzed thematically. The findings identify four primary forms of competency degradation: (1) deskilling, where fundamental technical skills erode through repeated cognitive delegation to AI; (2) cognitive passivity, where developers cease active algorithmic reasoning especially in exploration mode usage; (3) silent unlearning, an imperceptible and gradual loss of competency particularly in experienced developers; and (4) misleading sense of competence, where successful AI-assisted code generation creates a false perception of technical ability. These risks are modulated by five factors: developer experience level, AI usage mode, project scale and complexity, output verification practices, and the availability of formal training. Furthermore, the findings reveal that competency degradation constitutes not merely a technical issue but an ethical violation of professional standards established by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE), both of which mandate continuous technical competency maintenance throughout a professional career. This study concludes that AI code assistants must be adopted critically and reflectively, preserving the active role of developers in understanding, evaluating, and validating generated code to uphold both technical competence and professional responsibility.*

Keywords: *AI code assistant, competency degradation, deskilling, developer, professional ethics*

Abstrak. Adopsi luas *AI code assistant* seperti GitHub Copilot, ChatGPT, dan Amazon CodeWhisperer telah mentransformasi alur kerja pengembangan perangkat lunak secara signifikan. Meskipun alat-alat ini menawarkan peningkatan produktivitas yang terdokumentasi, termasuk penyelesaian tugas hingga 55% lebih cepat, penggunaannya yang tidak kritis menimbulkan risiko serius terhadap kompetensi teknis *developer* yang masih kurang dikaji dalam literatur. Penelitian ini bertujuan menganalisis risiko degradasi kompetensi teknis *developer* akibat penggunaan *AI code assistant* melalui tinjauan kepustakaan sistematis. Literatur dikumpulkan dari ScienceDirect, Scopus, dan Google Scholar dalam rentang 2020–2026, diseleksi menggunakan kriteria inklusi dan eksklusi yang telah ditetapkan, dan dianalisis secara tematik. Temuan mengidentifikasi empat bentuk utama degradasi kompetensi: (1) *deskilling*, yaitu erosi kemampuan teknis fundamental akibat delegasi kognitif berulang kepada AI; (2) *cognitive passivity*, kondisi *developer* berhenti melakukan penalaran algoritmik aktif terutama dalam mode eksplorasi; (3) *silent unlearning*, hilangnya kompetensi secara diam-diam terutama pada *developer* berpengalaman; dan (4) *misleading sense of competence*, ilusi kompetensi akibat keberhasilan menghasilkan kode dengan bantuan AI. Risiko ini dipengaruhi oleh lima faktor: tingkat pengalaman *developer*, mode penggunaan AI, skala dan kompleksitas proyek, praktik verifikasi *output*, serta ketersediaan pelatihan formal. Temuan juga menunjukkan bahwa degradasi kompetensi bukan sekadar masalah teknis, melainkan berimplikasi pada pelanggaran standar etika profesi yang ditetapkan ACM dan IEEE. Penelitian ini menyimpulkan bahwa *AI*

Naskah Masuk: 03 Juni 2026; Revisi: 04 Juni 2026; Diterima: 05 Juni 2026; ; Terbit: 06 Juni 2026.

code assistant harus diadopsi secara kritis dan reflektif demi menjaga kompetensi teknis dan tanggung jawab profesional.

Kata kunci: *AI code assistant*, degradasi kompetensi, *deskilling*, *developer*, etika profesi

1. LATAR BELAKANG

Dalam beberapa tahun terakhir, penggunaan kecerdasan buatan (*Artificial Intelligence*/AI) sebagai *code assistant* dalam proses pengembangan perangkat lunak mengalami peningkatan yang signifikan. *Tools* seperti GitHub Copilot, ChatGPT, dan Amazon CodeWhisperer kini telah menjadi bagian dari alur kerja sehari-hari para *developer* di seluruh dunia, didorong oleh kemampuannya dalam mengotomasi penulisan kode, memberikan saran secara *real-time*, serta mendukung berbagai tahapan pengembangan mulai dari penulisan fungsi, *debugging*, hingga pembuatan dokumentasi otomatis [1]. Adopsi teknologi ini berlangsung sangat cepat, dengan lebih dari 90% *developer* profesional kini menggunakan setidaknya satu *AI code assistant* dalam alur kerja mereka [3]. Kemampuan alat-alat ini bertumpu pada arsitektur *transformer* dan *large language model* (LLM) yang dilatih pada korpus kode sumber bervolume sangat besar, memungkinkan pembangkitan saran kode yang kontekstual dan adaptif [6]. Tren adopsi yang masif ini telah mengubah cara *developer* berinteraksi dengan kode, dan secara fundamental menggeser ekspektasi terhadap kompetensi teknis yang dibutuhkan dalam industri pengembangan perangkat lunak [1][8].

Penggunaan *AI code assistant* terbukti secara empiris mampu meningkatkan produktivitas *developer* secara terukur. Penelitian yang dilakukan oleh Alenezi dan Akour (2025) terhadap 250 profesional perangkat lunak menunjukkan bahwa 74% responden menyatakan AI mengurangi waktu yang diperlukan untuk tugas pengkodean rutin, sementara 62% mencatat penurunan jumlah *defect* pasca rilis [1]. Studi GitHub yang dikutip oleh Patel dan Patil (2025) menemukan bahwa *developer* yang menggunakan GitHub Copilot mampu menyelesaikan tugas pemrograman hingga 55% lebih cepat dibandingkan kelompok yang tidak menggunakannya, dengan nilai $p<0.05$ berdasarkan uji T-test [4][8]. Tinjauan sistematis terhadap 60 publikasi ilmiah yang dilakukan oleh Kozub et al. (2025) juga mengonfirmasi bahwa penggunaan *tools* AI generatif dapat menghasilkan reduksi waktu pengembangan sebesar 20-40% [10].

Manfaat-manfaat ini menjadikan *AI code assistant* sebagai alat yang semakin sulit dipisahkan dari praktik pengembangan perangkat lunak modern.

Namun di sisi lain, penggunaan *AI code assistant* yang tidak terkendali memunculkan kekhawatiran serius terhadap kompetensi teknis *developer* yang belum banyak dikaji dalam literatur. Nyembo Mpampi et al. (2025) menemukan bahwa lebih dari 60% mahasiswa menerima saran kode dari AI tanpa modifikasi, bahkan ketika kode tersebut mengandung kesalahan logis, menandakan adanya *cognitive passivity* yang mengikis kemampuan pemecahan masalah secara mandiri [2]. Glas et al. (2026) mengidentifikasi fenomena *misleading sense of competence*, yaitu kondisi di mana *developer* merasa kompeten karena berhasil menghasilkan kode yang berfungsi dengan bantuan AI, padahal pemahaman teknis mendasarnya tidak berkembang [3]. Lebih lanjut, Barke, James, dan Polikarpova (2024) menemukan bahwa penggunaan AI dalam mode eksplorasi secara langsung menghambat perkembangan kompetensi karena proses pembelajaran aktif tidak terjadi [7]. Fenomena-fenomena ini secara kolektif menggambarkan ancaman yang disebut sebagai *deskilling*, yaitu penurunan kemampuan teknis fundamental *developer* akibat ketergantungan berlebihan pada sistem AI [2].

Meskipun sejumlah penelitian telah membahas dampak AI terhadap produktivitas *developer* dan perubahan peran profesional, kajian yang secara sistematis menganalisis risiko degradasi kompetensi teknis dari perspektif etika profesi masih sangat terbatas [1][5]. Sebagian besar studi yang ada berfokus pada manfaat produktivitas atau aspek keamanan kode, tanpa mengkaji secara mendalam implikasi etika profesional dari fenomena degradasi kompetensi yang ditimbulkan [3][10]. Standar etika profesi yang ditetapkan oleh *Association for Computing Machinery* (ACM) dan *Institute of Electrical and Electronics Engineers* (IEEE) secara eksplisit mewajibkan para profesional teknologi informasi untuk senantiasa menjaga dan meningkatkan kompetensi teknisnya sepanjang karier [3]. Kesenjangan antara kewajiban etika profesi ini dengan realita degradasi kompetensi yang terjadi di lapangan merupakan gap penelitian yang mendesak untuk dijawab melalui kajian yang sistematis dan komprehensif.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk menganalisis risiko degradasi kompetensi teknis *developer* akibat penggunaan *AI code assistant* melalui pendekatan tinjauan kepustakaan sistematis. Penelitian ini mengajukan dua

pertanyaan utama: pertama, apa saja bentuk risiko degradasi kompetensi teknis *developer* yang teridentifikasi dalam literatur akibat penggunaan *AI code assistant*? Kedua, bagaimana perspektif etika profesi IT menyikapi fenomena tersebut? Kontribusi penelitian ini mencakup tiga hal utama: sintesis literatur yang terstruktur mengenai bentuk-bentuk degradasi kompetensi teknis *developer* akibat AI, kerangka analisis faktor-faktor yang memengaruhi tingkat degradasi yang dapat digunakan oleh organisasi dan institusi pendidikan, serta rekomendasi praktis berbasis etika profesi bagi *developer* dan institusi pendidikan teknologi informasi di Indonesia [1][2][3].

2. KAJIAN TEORITIS

AI Code Assistant dalam Pengembangan Perangkat Lunak

AI code assistant merupakan alat berbasis kecerdasan buatan yang dirancang untuk membantu *developer* dalam berbagai aspek pengembangan perangkat lunak. Alat-alat seperti GitHub Copilot, ChatGPT, Amazon CodeWhisperer, dan Tabnine memanfaatkan arsitektur *transformer* dan *large language model* (LLM) yang dilatih pada korpus kode sumber bervolume sangat besar untuk menghasilkan saran kode secara kontekstual [6]. Sistem ini bekerja melalui pemahaman konteks kode yang sedang ditulis, pembangkitan saran yang relevan, serta penyempurnaan saran secara adaptif [6]. Vaithilingam, Zhang, dan Glassman (2022) melalui studi pengguna terhadap 24 partisipan menemukan bahwa meskipun *Copilot* tidak selalu meningkatkan waktu penyelesaian tugas atau tingkat keberhasilan, sebagian besar peserta lebih memilih menggunakannya dalam pekerjaan sehari-hari karena menyediakan titik awal yang berguna. Namun, peserta mengalami kesulitan dalam memahami, mengedit, dan *debugging* kode yang dihasilkan oleh *Copilot*, yang secara signifikan menghambat efektivitas penyelesaian tugas [19].

Dari sisi manfaat, adopsi *AI code assistant* telah menghasilkan peningkatan produktivitas yang signifikan. Alenezi dan Akour (2025) melaporkan bahwa *developer* pengguna AI mampu menyelesaikan tugas pemrograman hingga 55% lebih cepat, dengan 74% responden menyatakan AI mengurangi waktu yang diperlukan untuk tugas pengkodean rutin [1]. Peng et al. (2023) mengonfirmasi melalui studi lapangan terkontrol bahwa penggunaan GitHub Copilot menghasilkan peningkatan signifikan dalam kecepatan dan kualitas kode [13]. Kozub et al. (2025) melalui tinjauan sistematis terhadap 60 publikasi menemukan bahwa penggunaan *tools* AI generatif dapat menghasilkan

reduksi waktu pengembangan sebesar 20-40% [10]. Sauvola et al. (2024) menambahkan bahwa AI generatif berpotensi mengubah paradigma pengembangan perangkat lunak secara mendasar dengan memungkinkan otomasi tugas di seluruh tahapan SDLC, meskipun juga membawa risiko dan tantangan baru yang membutuhkan pendekatan tata kelola yang komprehensif [20].

Konsep Deskilling dan Degradasi Kompetensi Akibat Otomasi Teknologi

Konsep *deskilling* dalam konteks teknologi memiliki akar teoritis yang panjang. Rafner et al. (2022) mendefinisikan *deskilling* sebagai proses di mana otomasi teknologi secara bertahap menggantikan atau melemahkan keahlian yang sebelumnya dimiliki oleh manusia, sambil secara bersamaan menciptakan tuntutan kompetensi baru yang disebut *upskilling* dan *reskilling* [16]. Crowston dan Bolici (2025) memperkuat konsep ini dalam konteks AI, menemukan bahwa meskipun AI memunculkan *upskilling* di area baru seperti *prompt engineering*, kompetensi teknis inti seperti penalaran algoritmik dan *debugging* mandiri berisiko tererosi secara signifikan [14]. Prather et al. (2023) mengidentifikasi pola interaksi bermasalah antara pemrogram pemula dengan *Copilot*, termasuk *shepherding* di mana *developer* mengetikkan saran AI karakter demi karakter tanpa benar-benar memahaminya, dan *drifting* di mana *developer* menerima kode AI yang salah dan tersesat dalam proses *debugging* yang tidak produktif [22].

Pearce et al. (2022) melalui analisis sistematis terhadap 1.689 program yang dihasilkan oleh GitHub Copilot dalam 89 skenario berbeda menemukan bahwa sekitar 40% dari program tersebut mengandung kerentanan keamanan [24]. Temuan ini menjadi bukti teknis yang kuat bahwa ketergantungan pada *output* AI tanpa verifikasi memadai membawa risiko nyata, sebuah kondisi yang langsung berkaitan dengan degradasi kemampuan *developer* dalam mendeteksi kerentanan keamanan. Russo (2024) dalam studi tentang adopsi AI generatif dalam rekayasa perangkat lunak menemukan bahwa ketergantungan yang tumbuh pada alat AI menciptakan tantangan serius bagi pemeliharaan kompetensi tim, terutama dalam konteks proyek yang kompleks dan berubah cepat [15]. Glas et al. (2026) menambahkan dimensi *misleading sense of competence*, di mana keberhasilan menghasilkan kode yang berfungsi menciptakan ilusi kompetensi yang memperlemah kemampuan *developer* dalam mendeteksi kerentanan keamanan [3].

Etika Profesi dalam Teknologi Informasi

Etika profesi teknologi informasi diatur oleh dua standar internasional utama: ACM *Code of Ethics and Professional Conduct* (2018) dan IEEE *Code of Ethics*. Kedua standar ini secara eksplisit mengamanatkan bahwa profesional TI memiliki kewajiban untuk senantiasa menjaga dan meningkatkan kompetensi teknisnya sepanjang karier [3]. Atemkeng et al. (2024) dalam kajiannya tentang etika pemrograman di era AI generatif menegaskan bahwa penggunaan AI dalam pemrograman memunculkan tantangan etis fundamental yang belum sepenuhnya dijawab oleh kerangka etika profesi yang ada, termasuk persoalan akuntabilitas, kepemilikan intelektual, dan pemeliharaan kompetensi [17]. Klemmer et al. (2024) dalam studi kualitatif terhadap 27 profesional menemukan bahwa *developer* melaporkan kekhawatiran tentang masa depan profesi mereka akibat ketergantungan pada AI, namun kebijakan organisasional yang mengatur penggunaan AI dalam pengembangan perangkat lunak yang aman masih sangat terbatas [23].

Penelitian empiris Abubakar, Jeilani, dan Yusuf (2025) memberikan bukti bahwa kesadaran etis dapat berfungsi sebagai moderator efektif terhadap dampak negatif ketergantungan AI (b=0,0978, p=0,031), dan kebijakan institusional yang kuat berperan sebagai penyeimbang (b=0,115, p=0,035) [9]. Perry et al. (2023) menambahkan bahwa *developer* yang memiliki akses ke AI *assistant* lebih cenderung percaya bahwa mereka telah menulis kode yang aman meskipun kenyataannya sebaliknya — sebuah temuan yang secara langsung menunjukkan bagaimana ketergantungan AI dapat melemahkan penilaian etis dan profesional *developer* [18]. Kajian lintas literatur menunjukkan bahwa dimensi etika dalam penggunaan AI, mencakup *intellectual property*, privasi data, dan *ethical governance*, belum ditangani secara memadai oleh komunitas peneliti maupun industri [5][10].

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan studi kepustakaan sistematis (*systematic literature review*). Studi kepustakaan merupakan studi yang digunakan dalam mengumpulkan informasi dan data dengan bantuan berbagai macam material yang ada di perpustakaan seperti dokumen, buku, majalah, kisah-kisah sejarah, dan sebagainya. Studi kepustakaan juga berarti teknik pengumpulan data dengan melakukan penelaahan terhadap buku, literatur, catatan, serta berbagai laporan yang berkaitan dengan masalah

yang ingin dipecahkan (Nazir, 1988 dalam Mirzaqon, 2017). Menurut ahli lain, studi kepustakaan merupakan kajian teoritis, referensi serta literatur ilmiah lainnya yang berkaitan dengan budaya, nilai dan norma yang berkembang pada situasi sosial yang diteliti (Sugiyono, 2012 dalam Mirzaqon, 2017). Pendekatan sistematis dalam penelitian ini berarti proses pencarian, seleksi, dan analisis literatur dilakukan secara terstruktur dan terdokumentasi sehingga dapat direplikasi oleh peneliti lain.

Pencarian literatur dilakukan pada beberapa sumber ilmiah, yaitu ScienceDirect, Scopus, dan Google Scholar. Ketiga sumber ini dipilih karena mencakup jurnal-jurnal bereputasi tinggi di bidang ilmu komputer, rekayasa perangkat lunak, dan sistem informasi, sekaligus memungkinkan akses terhadap artikelartikel jurnal internasional yang relevan secara lebih luas. Pencarian dilakukan menggunakan kombinasi kata kunci yang disusun berdasarkan tiga konsep utama penelitian ini, yaitu *AI code assistant*, degradasi kompetensi atau *deskilling*, serta *developer* atau rekayasa perangkat lunak. Kombinasi kata kunci yang digunakan antara lain "*AI code assistant* AND *skill degradation* OR *deskilling* AND *software developer*", "*AI-assisted development* AND *over-reliance* OR *cognitive passivity* AND *programmer*", serta "*generative AI* AND *software engineering* AND *professional ethics* OR *developer competency*". Pencarian dibatasi pada artikel yang diterbitkan antara tahun 2020 hingga 2026 untuk memastikan relevansi dengan perkembangan *AI code assistant* generasi terkini.

Proses seleksi artikel dilakukan berdasarkan kriteria inklusi dan eksklusi yang telah ditetapkan sebelum pencarian dilakukan. Artikel dimasukkan ke dalam kajian apabila memenuhi seluruh kriteria berikut: pertama, membahas dampak penggunaan *AI code assistant* terhadap kompetensi teknis, kemampuan algoritmik, atau perilaku profesional *developer*; kedua, diterbitkan dalam rentang tahun 2020 hingga 2026; ketiga, ditulis dalam bahasa Inggris atau Indonesia; keempat, diterbitkan di jurnal atau prosiding yang terindeks Scopus, ScienceDirect, IEEE Xplore, atau ACM Digital Library; serta kelima, tersedia dalam versi *full-text*. Sebaliknya, artikel dikeluarkan dari kajian apabila memenuhi salah satu kriteria berikut: pertama, hanya membahas produktivitas atau kecepatan *developer* tanpa menyentuh aspek kompetensi atau degradasi kemampuan; kedua, membahas penerapan AI di luar domain pengembangan perangkat lunak; ketiga,

merupakan artikel opini atau editorial tanpa analisis sistematis; keempat, tidak tersedia dalam versi *full-text*; atau kelima, merupakan duplikat dari artikel yang telah dimasukkan.

Analisis terhadap literatur yang terpilih dilakukan melalui pendekatan tematik, yaitu dengan mengidentifikasi, mengelompokkan, dan mensintesis temuan-temuan utama dari setiap artikel ke dalam tema-tema yang relevan dengan pertanyaan penelitian. Tema-tema yang dihasilkan kemudian disajikan secara sistematis pada bagian hasil dan pembahasan untuk menjawab dua pertanyaan penelitian yang telah dirumuskan. Seluruh proses analisis dilakukan secara iteratif hingga tercapai saturasi tema, yaitu kondisi di mana penambahan literatur baru tidak lagi menghasilkan tema atau dimensi analisis yang baru.

4. HASIL DAN PEMBAHASAN (Sub judul level 1)

Gambaran Penggunaan AI Code Assistant dalam Pengembangan Perangkat Lunak

Penggunaan *AI code assistant* dalam proses pengembangan perangkat lunak telah berkembang pesat dalam beberapa tahun terakhir. Alat-alat seperti GitHub Copilot, ChatGPT, Amazon CodeWhisperer, dan Tabnine kini digunakan secara luas di berbagai fase SDLC, mulai dari analisis kebutuhan, penulisan kode, pengujian, hingga *deployment* dan pemeliharaan sistem [5]. Liang, Yang, dan Myers (2024) melalui survei terhadap 410 *developer* menemukan bahwa motivasi utama penggunaan AI *programming assistant* adalah pengurangan penekanan tombol, penyelesaian tugas yang lebih cepat, dan pengingatan sintaksis, meskipun *developer* kurang tertarik menggunakannya untuk *brainstorming* solusi [21]. Analisis terhadap 112 responden profesional oleh Anitha et al. (2025) menunjukkan bahwa fase pengujian mendapatkan skor dampak AI tertinggi sebesar 10, diikuti fase pengkodean sebesar 9, pengumpulan kebutuhan sebesar 8, dan *deployment* sebesar 7 [8].

Penggunaan *AI code assistant* membawa manfaat produktivitas yang tidak dapat diabaikan. *Developer* pengguna AI mampu menyelesaikan tugas pemrograman hingga 55% lebih cepat dibandingkan non-pengguna [13], dan uji T-test mengonfirmasi perbedaan signifikan dalam efisiensi pengkodean antara kedua kelompok dengan nilai $p<0.05$ [8]. Moradi Dakhel et al. (2023) melalui analisis komparatif menemukan bahwa *developer* yang menggunakan GitHub Copilot menghasilkan kode dengan tingkat

kelulusan *test case* yang lebih tinggi [12]. Meski demikian, justru di balik manfaat produktivitas inilah risiko degradasi kompetensi tersembunyi dan berkembang secara diam-diam. Vaithilingam, Zhang, dan Glassman (2022) memperingatkan bahwa kesulitan dalam memahami, mengedit, dan melakukan *debugging* kode yang dihasilkan oleh AI merupakan tantangan usabilitas yang signifikan yang dapat berdampak negatif pada kompetensi jangka panjang *developer* [19].

Bentuk-Bentuk Risiko Degradasi Kompetensi Teknis Developer

Literatur mengidentifikasi setidaknya ada empat bentuk utama degradasi kompetensi teknis *developer* akibat penggunaan *AI code assistant* yang tidak kritis. Keempat bentuk ini saling berkaitan dan dapat terjadi secara bersamaan, terutama pada *developer* yang menggunakan AI tanpa pemahaman kritis terhadap *output*-nya.

Bentuk pertama adalah *deskilling* atau penurunan kemampuan teknis secara bertahap. Ketika *developer* secara sistematis mendelegasikan tugas kognitif kepada AI, kemampuan teknis fundamentalnya, seperti penalaran algoritmik, pemodelan solusi, dan *debugging* mandiri berangsur melemah [2]. Lebih dari 60% mahasiswa terbukti menerima saran kode dari AI tanpa modifikasi, bahkan ketika kode tersebut mengandung kesalahan logis atau implementasi yang tidak efisien [2]. Crowston dan Bolici (2025) menemukan bahwa meskipun AI menciptakan peluang untuk *upskilling* di area baru, kompetensi teknis inti seperti kemampuan membangun solusi algoritmik dari nol berisiko tererosi secara signifikan [14]. Rafner et al. (2022) menegaskan bahwa *deskilling* akibat otomasi adalah fenomena yang terdokumentasi dengan baik dalam sejarah teknologi, dan AI *code assistant* merepresentasikan iterasi terbaru dari pola yang sama dalam konteks rekayasa perangkat lunak [16].

Bentuk kedua adalah *cognitive passivity* atau pasivitas kognitif. Penelitian Barke, James, dan Polikarpova (2024) menemukan dua mode penggunaan AI yang berbeda: mode akselerasi di mana *developer* menggunakan AI untuk mempercepat tugas yang sudah dipahami, dan mode eksplorasi di mana *developer* menggunakan AI untuk menyelesaikan tugas yang belum dipahami sepenuhnya [7]. Prather et al. (2023) memperkuat temuan ini dengan mengidentifikasi pola *shepherding* — di mana *developer* pemula mengetikkan saran AI karakter demi karakter tanpa memahami kode tersebut — dan *drifting* — di mana *developer* menerima kode AI yang salah dan tersesat dalam

debugging rabbit hole yang semakin menjauh dari solusi yang benar [22]. Liang, Yang, dan Myers (2024) mengonfirmasi bahwa dalam praktiknya *developer* tidak menerima saran awal AI *assistant* dengan frekuensi tinggi, menandakan adanya friksi kognitif dalam interaksi dengan AI yang dapat berdampak negatif pada produktivitas berpikir kritis [21].

Bentuk ketiga adalah *silent unlearning* atau penurunan kompetensi diam-diam. Berbeda dengan *deskilling* yang bersifat progresif, *silent unlearning* terjadi tanpa disadari oleh *developer* itu sendiri [2]. *Developer* berpengalaman yang terbiasa menyelesaikan masalah kompleks secara mandiri dapat kehilangan kemampuan tersebut secara perlahan ketika mereka mulai mendelegasikan tugas-tugas yang sama kepada AI. Dalam jangka panjang, kondisi ini berpotensi mengakibatkan penurunan kapasitas inovasi algoritmik dan meningkatnya ketergantungan pada alat *proprietary* yang mengurangi otonomi teknis *developer* [2]. Kozub et al. (2025) mengidentifikasi bahwa dampak AI pada dinamika tim dan distribusi peran masih sangat minim diteliti, mengindikasikan bahwa *silent unlearning* pada level organisasional merupakan area yang memerlukan perhatian lebih mendalam [10].

Bentuk keempat adalah *misleading sense of competence* atau ilusi kompetensi. Ini merupakan bentuk degradasi yang paling berbahaya karena tidak terlihat dari luar [3]. *Developer*, terutama yang kurang berpengalaman, merasa kompeten karena berhasil menghasilkan kode yang berfungsi dengan bantuan AI, padahal pemahaman teknis mendasarnya tidak berkembang [3]. Perry et al. (2023) memberikan bukti empiris yang kuat: peserta yang menggunakan AI *assistant* tidak hanya menghasilkan kode yang lebih tidak aman, tetapi juga secara signifikan lebih cenderung menilai kode tidak aman mereka sebagai kode yang aman dibandingkan kelompok kontrol [18]. Glas et al. (2026) menemukan bahwa bahkan *developer* berpengalaman pun mengalami *overtrust* terhadap *output* AI, di mana mereka cenderung menganggap kode yang dihasilkan AI lebih aman atau berkualitas lebih tinggi dibandingkan kode yang ditulis manusia [3]. Keempat bentuk degradasi kompetensi disajikan dalam Tabel 1.

Bentuk Degradasi	Definisi	Mekanisme Utama
Deskilling	Penurunan kemampuan teknis secara bertahap	Delegasi tugas kognitif berulang ke AI
Cognitive Passivity	Pasivitas dalam proses berpikir algoritmik	*Developer* menerima saran AI tanpa evaluasi kritis
Silent Unlearning	Penurunan kompetensi yang tidak disadari	Kemampuan memudar karena tidak dipraktikkan mandiri
Misleading Sense of Competence	Ilusi kompetensi akibat keberhasilan semu	Kode berfungsi karena AI, bukan pemahaman *developer*

Faktor-Faktor yang Memengaruhi Tingkat Degradasi Kompetensi

Tingkat degradasi kompetensi yang dialami seorang *developer* tidak bersifat seragam. Terdapat beberapa faktor yang menentukan seberapa parah dampak yang ditimbulkan. Literatur mengidentifikasi lima faktor utama yang memengaruhi tingkat degradasi kompetensi *developer* akibat penggunaan *AI code assistant*.

Faktor pertama adalah tingkat pengalaman *developer*. *Developer* pemula yang belum memiliki fondasi kompetensi teknis yang kuat lebih rentan mengalami *deskilling* dan *misleading sense of competence*, karena mereka belum mampu mengevaluasi kualitas *output* AI secara kritis [2][3]. Prather et al. (2023) menemukan bahwa pemrogram pemula sering kali menerima saran AI yang salah tanpa menyadarinya, yang langsung menghambat perkembangan pemahaman teknis mendasar mereka [22]. Sebaliknya, *developer* berpengalaman lebih rentan terhadap *silent unlearning* karena mereka memiliki kemampuan evaluasi terhadap AI, namun secara bertahap berhenti melakukannya karena terbiasa mempercayai *output* AI [2].

Faktor kedua adalah mode penggunaan AI. Penelitian menemukan bahwa cara *developer* berinteraksi dengan *AI code assistant* sangat menentukan tingkat risiko degradasi yang terjadi [7]. Penggunaan AI dalam mode akselerasi, yaitu kondisi di mana

developer menggunakan AI untuk mempercepat tugas yang sudah dipahami dengan baik, relatif aman dan tidak memicu degradasi kompetensi.

Sebaliknya, penggunaan AI dalam mode eksplorasi, yaitu kondisi di mana AI digunakan untuk menyelesaikan masalah yang belum dipahami, secara langsung menghambat perkembangan kompetensi karena proses pembelajaran aktif tidak terjadi [7]. Liang, Yang, dan Myers (2024) mengonfirmasi bahwa ketidakmampuan *developer* untuk secara konsisten mengenali kapan AI memberikan saran yang tepat merupakan salah satu tantangan usabilitas terbesar yang berdampak pada efektivitas penggunaan [21].

Faktor ketiga adalah skala dan kompleksitas proyek. AI menunjukkan efektivitas tertinggi pada tugas yang terlokalisasi dan terdefinisi dengan baik seperti penulisan fungsi standar dan *unit test*. Akan tetapi, AI sering kali gagal mempertahankan koherensi pada proyek berskala besar yang melibatkan dependensi antarkomponen yang kompleks [1][5]. Pearce et al. (2022) menemukan bahwa kualitas dan keamanan kode yang dihasilkan GitHub Copilot sangat bervariasi tergantung pada konteks dan domain, dengan sekitar 40% program yang dihasilkan mengandung kerentanan keamanan [24]. *Developer* yang terlalu bergantung pada AI dalam proyek kompleks justru kehilangan kesempatan untuk membangun kompetensi arsitektur sistem dan pemecahan masalah lintas modul yang tidak dapat dipelajari dari AI [1].

Faktor keempat adalah tingkat verifikasi *output*. *Developer* yang secara konsisten memverifikasi dan mempertanyakan setiap saran yang dihasilkan AI terbukti lebih mampu mempertahankan kompetensi teknisnya [3][8]. Perry et al. (2023) menemukan bahwa banyak partisipan eksperimen menerima solusi AI sebagai jawaban akhir mereka tanpa verifikasi tambahan, bahkan ketika solusi tersebut mengandung kerentanan keamanan yang signifikan [18]. Klemmer et al. (2024) mengonfirmasi bahwa *developer* sering kali menilai *output* AI terlalu optimis, dan kebiasaan verifikasi yang kritis merupakan pembeda utama antara *developer* yang mampu mempertahankan kompetensi dan yang mengalami degradasi [23].

Faktor kelima adalah ketersediaan pelatihan formal. Tim yang menerima pelatihan AI formal mencapai peningkatan efisiensi sebesar 30% lebih tinggi dibandingkan dengan tim yang tidak mendapatkan pelatihan [1]. Abubakar et al. (2025) mengonfirmasi bahwa

kebijakan institusional yang kuat berperan sebagai penyeimbang yang efektif terhadap dampak negatif ketergantungan AI (b=0,115, p=0,035) [9]. Rafner et al. (2022) menegaskan bahwa program *reskilling* yang terstruktur merupakan komponen esensial dalam strategi organisasi untuk menghadapi dampak *deskilling* akibat otomasi berbasis AI [16].

Implikasi Etika Profesi terhadap Fenomena Degradasi Kompetensi

Fenomena degradasi kompetensi yang teridentifikasi dalam literatur membawa implikasi langsung terhadap etika profesi teknologi informasi. Standar etika profesi yang ditetapkan oleh ACM dan IEEE secara eksplisit mengamanatkan bahwa profesional TI memiliki kewajiban untuk senantiasa menjaga dan meningkatkan kompetensi teknisnya sepanjang karier. Ketika penggunaan *AI code assistant* secara sistematis mengikis kompetensi tersebut melalui *deskilling*, *cognitive passivity*, *silent unlearning*, dan *misleading sense of competence*, maka *developer* tidak hanya menghadapi risiko teknis, tetapi juga berpotensi melanggar prinsip etika profesi yang mendasar [3]. Atemkeng et al. (2024) menegaskan bahwa era AI generatif membutuhkan reinterpretasi dan penguatan standar etika profesi yang ada, karena banyak tantangan etika yang muncul belum sepenuhnya diantisipasi oleh kerangka ACM dan IEEE [17].

Implikasi etika ini semakin serius ketika dikaitkan dengan tanggung jawab *developer* atas keamanan perangkat lunak yang dihasilkan. Glas et al. (2026) menemukan bahwa AI tidak secara *default* menghasilkan kode yang aman karena dilatih pada data sumber terbuka yang kualitasnya bervariasi [3]. Pearce et al. (2022) membuktikan secara empiris bahwa sekitar 40% dari kode yang dihasilkan GitHub Copilot dalam skenario keamanan mengandung kerentanan yang dapat dieksploitasi [24]. Perry et al. (2023) menambahkan bahwa *developer* yang menggunakan AI lebih cenderung percaya bahwa kode mereka aman meskipun kenyataannya tidak demikian, sebuah kondisi yang secara langsung merupakan manifestasi *misleading sense of competence* yang berimplikasi etis serius [18]. Klemmer et al. (2024) menemukan bahwa bahkan *developer* berpengalaman pun cenderung *overtrust* terhadap kode yang dihasilkan AI dalam konteks keamanan perangkat lunak [23].

Literatur merumuskan tiga strategi utama untuk menjaga kompetensi *developer* sekaligus memenuhi kewajiban etika profesi di era AI. Pertama, mempertahankan *manual*

coding sebagai fondasi pembelajaran, khususnya di tahap awal karier [2]. Kedua, mengintegrasikan AI secara kritis dan reflektif dengan tidak menerima saran AI begitu saja, melainkan secara aktif mempertanyakan relevansi, keamanan, dan efisiensinya [2][3]. Ketiga, membangun budaya kolaborasi manusia-mesin di mana *developer* tetap menjadi validator akhir dari seluruh kode yang digunakan dalam produksi [1]. Rafner et al. (2022) menambahkan bahwa pendekatan *hybrid intelligence*, di mana kemampuan manusia dan AI saling melengkapi tanpa menggantikan satu sama lain, merupakan model yang paling etis dan berkelanjutan [16].

Perspektif etika profesi terhadap penggunaan AI juga mendapat dukungan dari kajian lintas domain teknologi informasi. Nurmiati dan Rizqi (2026) dalam tinjauan sistematis mereka menemukan bahwa penyalahgunaan teknologi AI melanggar prinsip-prinsip utama dalam ACM *Code of Ethics* dan IEEE *Code of Ethics* yang sama, khususnya prinsip *non-maleficence*, akuntabilitas, dan tanggung jawab sosial [26]. Relevansi temuan ini bagi konteks penelitian adalah bahwa kegagalan developer dalam memverifikasi dan memahami kode yang dihasilkan AI code assistant pada akhirnya berisiko menghasilkan perangkat lunak yang merugikan pengguna, sebuah kondisi yang secara langsung bertentangan dengan kewajiban etika profesi yang melekat pada setiap profesional teknologi informasi.

Perspektif Lintas Konteks: Ketergantungan AI dalam Pendidikan dan Implikasinya

Temuan-temuan dalam literatur terkait degradasi kompetensi teknis *developer* memperoleh penguatan dari penelitian empiris yang dilakukan dalam konteks pendidikan tinggi. Abubakar, Jeilani, dan Yusuf (2025) dalam kajiannya terhadap 226 mahasiswa di Universitas Mogadishu menemukan bahwa ketergantungan berlebih pada AI secara signifikan berkorelasi positif dengan konsekuensi negatif pada kemampuan belajar ($b=0,207$, $p=0,001$), bahwa kekhawatiran etis bertindak sebagai moderator yang signifikan dalam melemahkan dampak negatif tersebut ($b=0,0978$, $p=0,031$), dan bahwa kebijakan institusional yang kuat turut berperan sebagai penyeimbang yang efektif ($b=0,115$, $p=0,035$) [9]. Temuan ini dihasilkan melalui analisis regresi linear berganda dan pendekatan moderasi yang memberikan bukti statistik yang kuat tentang mekanisme psikologis yang mendasari ketergantungan AI.

Paralel yang paling relevan dengan konteks penelitian ini terletak pada mekanisme kognitif yang mendasari fenomena ketergantungan AI. Mahasiswa yang mengalami ketergantungan berlebih pada AI cenderung mengurangi keterlibatan berpikir kritis dan motivasi untuk menganalisis materi secara mandiri Hal ini menunjukkan sebuah pola yang dalam konteks pengembangan perangkat lunak disebut sebagai *cognitive passivity* dan *silent unlearning* [9]. Temuan bahwa semakin tinggi kesadaran etis, semakin berkurang dampak negatif dari ketergantungan AI, memiliki implikasi langsung terhadap pentingnya pembinaan budaya etika profesi dalam komunitas *developer* [9]. Standar ACM dan IEEE yang mewajibkan pemeliharaan kompetensi teknis bukan sekadar regulasi formal, melainkan instrumen kesadaran etis yang apabila diinternalisasi secara mendalam oleh *developer* dapat berfungsi sebagai pelindung terhadap kecenderungan *over-reliance* pada AI

5. KESIMPULAN DAN SARAN

Kesimpulan

Penelitian ini telah melakukan tinjauan kepustakaan sistematis terhadap literatur yang membahas dampak penggunaan *AI code assistant* terhadap kompetensi teknis *developer* dalam proses pengembangan perangkat lunak. Berdasarkan sintesis dari berbagai literatur yang dikaji, dua pertanyaan penelitian yang diajukan dapat dijawab sebagai berikut.

Pertama, terdapat empat bentuk utama risiko degradasi kompetensi teknis *developer* yang teridentifikasi dalam literatur akibat penggunaan *AI code assistant* secara tidak kritis. *Deskilling* terjadi ketika delegasi tugas kognitif secara berulang kepada AI menyebabkan kemampuan teknis fundamental *developer* melemah secara bertahap, yang terbukti dialami oleh lebih dari 60% subjek penelitian yang menerima saran kode AI tanpa modifikasi. *Cognitive passivity* muncul ketika *developer* berhenti berperan aktif dalam proses perancangan algoritmik, khususnya dalam mode eksplorasi yang secara langsung mengikis kemampuan pemecahan masalah secara mandiri. *Silent unlearning* merupakan ancaman paling sulit dideteksi karena terjadi secara diam-diam pada *developer* berpengalaman yang secara perlahan kehilangan kemahiran yang pernah dimilikinya. Sementara itu, *misleading sense of competence* menjadi bentuk risiko yang paling berbahaya karena menciptakan ilusi kompetensi yang memperlemah kemampuan

developer dalam mendeteksi kerentanan keamanan dan memahami arsitektur sistem. Keempat bentuk degradasi ini saling berkaitan dan dipengaruhi oleh faktor pengalaman *developer*, mode penggunaan AI, skala proyek, tingkat verifikasi *output*, dan ketersediaan pelatihan formal.

Kedua, dari perspektif etika profesi teknologi informasi, fenomena degradasi kompetensi ini bukan sekadar persoalan teknis melainkan juga merupakan pelanggaran terhadap prinsip etika profesi yang ditetapkan oleh ACM dan IEEE. Kedua standar tersebut secara eksplisit mengamanatkan kewajiban profesional TI untuk senantiasa menjaga dan meningkatkan kompetensi teknisnya sepanjang karier. Ketika penggunaan *AI code assistant* secara sistematis mengikis kompetensi tersebut, *developer* tidak hanya menghadapi risiko penurunan kualitas perangkat lunak yang dihasilkan, tetapi juga berpotensi melanggar tanggung jawab etis profesionalnya. Dengan demikian, penggunaan *AI code assistant* yang bertanggung jawab mensyaratkan sikap kritis dan reflektif agar kompetensi manusia tetap menjadi inti dari setiap keputusan teknis dalam pengembangan perangkat lunak.

Saran

Bagi *developer* dan praktisi perangkat lunak, disarankan untuk menggunakan *AI code assistant* secara selektif dan kritis dengan membedakan antara mode akselerasi dan mode eksplorasi. AI sebaiknya digunakan untuk mempercepat tugas yang sudah dipahami dengan baik, bukan sebagai pengganti proses berpikir untuk masalah yang belum dipahami. Selain itu, *developer* disarankan untuk secara konsisten memverifikasi setiap *output* AI, mempertanyakan relevansi dan keamanannya, serta meluangkan waktu untuk berlatih *manual coding* secara berkala guna mencegah terjadinya *silent unlearning*.

Bagi institusi pendidikan teknologi informasi, disarankan untuk merancang kurikulum yang mengintegrasikan *AI code assistant* secara bertahap dan terkontrol. Program pembelajaran sebaiknya memastikan fondasi kompetensi teknis dibangun terlebih dahulu sebelum mahasiswa diperkenalkan dengan alat bantuan AI, serta mencakup latihan evaluasi kritis terhadap kode hasil AI sehingga mahasiswa mengembangkan kemampuan mendeteksi kesalahan dan kerentanan keamanan dalam *output* AI sejak dini.

Bagi peneliti selanjutnya, penelitian ini merekomendasikan beberapa arah penelitian yang belum terjawab secara memadai dalam literatur. Pertama, dibutuhkan penelitian empiris yang mengukur secara langsung tingkat degradasi kompetensi *developer* dalam konteks Indonesia, mengingat seluruh literatur yang ditinjau bersumber dari konteks internasional. Kedua, perlu dikembangkan instrumen pengukuran *deskilling* dan *cognitive passivity* yang terstandarisasi agar fenomena ini dapat diteliti secara lebih konsisten di berbagai konteks organisasi. Ketiga, penelitian tentang dampak jangka panjang penggunaan *AI code assistant* terhadap kompetensi *developer* dalam rentang waktu lima tahun atau lebih masih sangat terbatas dan perlu mendapat perhatian lebih besar dari komunitas peneliti sistem informasi dan rekayasa perangkat lunak.

DAFTAR REFERENSI

[1] M. Alenezi and M. Akour, "AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions," *Appl. Sci.*, vol. 15, no. 3, p. 1344, Jan. 2025, doi: 10.3390/app15031344.

[2] A. Nyembo Mpampi, R. Ilunga Kalanda, P. Mpemba Mukonkole, A. Ebambi Lukombe, G. Mulumba Mulondo, and C. Ngoyi Tshite, "AI-Assisted Coding: Evolution or Erosion of Software Development Skills?," *Int. J. Math. Comput. Res.*, vol. 13, no. 9, pp. 5663–5671, Sep. 2025, doi: 10.47191/ijmcr/v13i9.10. [3] M. Glas, C. Nirschl, B. Lanyado, and J. van Niekerk, "Insecure by Design? A Human-Centric Security Perspective on AI-Assisted Software Development," *Comput. Secur.*, vol. 164, p. 104842, 2026, doi: 10.1016/j.cose.2026.104842.

[4] A. Patel and A. Patil, "The Impact of AI on Software Development: A Case Study on Copilot & ChatGPT," *Dept. of Computer Science & Engineering, PIT, Parul University*, Vadodara, India, 2025.

[5] A. Ochani, Y. Khaire, P. Gupta, P. Ingle, and S. Mishra, "AI-Powered Software Development: A Systematic Review of Artificial Intelligence in the Software Development Lifecycle," *Int. J. Res. Trends Innov.*, vol. 10, no. 10, pp. b174–b178, Oct. 2025.

[6] S. P. Velaga, "AI-Assisted Code Generation and Optimization: Leveraging Machine Learning to Enhance Software Development Processes," *Int. J. Innov. Eng. Res. Technol.*, vol. 7, no. 9, pp. 177–186, Sep. 2020.

[7] S. Barke, M. B. James, and N. Polikarpova, "Grounded Copilot: How Programmers Interact with CodeGenerating Models," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA1, Apr. 2023, doi: 10.1145/3586030.

[8] C. Anitha, N. K. Gupta, B. Chintala, D. Pilli, E. Kesavan, and S. M. S. Ali, "Impact of Artificial Intelligence on Software Development Processes," *J. Inf. Syst. Eng. Manag.*, vol. 10, no. 25s, pp. 431–437, 2025.

[9] S. Abubakar, A. Jeilani, and M. Yusuf, "The Role of Over-Reliance on AI in the Negative Consequences of Student Learning: The Moderating Effects of Ethical Concerns and Institutional Policies," *Cogent Educ.*, vol. 12, no. 1, p. 2591503, 2025, doi: 10.1080/2331186X.2025.2591503.

[10] V. Kozub, V. Druzhynin, D. Trufanova, P. Ihnatenko, and K. Kolos, "Using Artificial Intelligence in Software Development Processes: Achievements and Challenges," *Sustain. Eng. Innov.*, vol. 7, no. 2, pp. 463–476, Oct. 2025, doi: 10.37868/sei.v7i2.id526.

[11] Stack Overflow, "Stack Overflow Developer Survey 2025," Stack Overflow, 2025. [Online]. Available: https://survey.stackoverflow.co/2025/

[12] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, "GitHub Copilot AI Pair Programmer: Asset or Liability?," *J. Syst. Softw.*, vol. 203, p. 111734, Sep. 2023, doi: 10.1016/j.jss.2023.111734.

[13] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot," *arXiv preprint arXiv:2302.06590*, Feb. 2023, doi:10.48550/arXiv.2302.06590.

[14] K. Crowston and F. Bolici, "Deskilling and Upskilling with AI Systems," *Inf. Res.*, vol. 30, no. iConf, pp. 1009–1023, 2025.

[15] D. Russo, "Navigating the Complexity of Generative AI Adoption in Software Engineering," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 5, pp. 1–50, 2024, doi: 10.1145/3652154.

[16] J. Rafner, D. Dellermann, A. Hjorth, D. Verasztó, C. Kampf, W. Mackay, and J. Sherson, "Deskilling, Upskilling, and Reskilling: A Case for Hybrid Intelligence," *Morals Mach.*, vol. 1, no. 2, pp. 24–39, 2022, doi: 10.5771/2747-5174-2022-2-24.

[17] M. Atemkeng, S. Hamlomo, B. Welman, N. Oyetunji, P. Ataei, and J. L. K. E. Fendji, "Ethics of Software Programming with Generative AI: Is Programming without Generative AI Always Radical?," *arXiv preprint arXiv:2408.10554*, 2024, doi: 10.48550/arXiv.2408.10554.

[18] N. Perry, M. Srivastava, D. Kumar, and D. Boneh, "Do Users Write More Insecure Code with AI Assistants?," in *Proc. 2023 ACM SIGSAC Conf. Computer and Communications Security (CCS '23)*, Copenhagen, Denmark, Nov. 2023, pp. 2785–2799, doi: 10.1145/3576915.3623157.

[19] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models," in *CHI Conf. Human Factors in Computing Systems Extended Abstracts (CHI '22)*, New Orleans, LA, USA, Apr. 2022, doi: 10.1145/3491101.3519665.

[20] J. Sauvola, S. Tarkoma, M. Klemettinen, J. Riekki, and D. Doermann, "Future of Software Development with Generative AI," *Autom. Softw. Eng.*, vol. 31, no. 1, p. 26, Mar. 2024, doi: 10.1007/s10515-024-00426z.

[21] J. T. Liang, C. Yang, and B. A. Myers, "A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges," in *Proc. 46th IEEE/ACM*

Int. Conf. Software Engineering (ICSE 2024), Lisbon, Portugal, Apr. 2024, pp. 616–628, doi: 10.1145/3597503.3608128.

[22] J. Prather, B. Reeves, K. Denny, E. Becker, and D. Leinonen, "It's Weird That It Knows What I Want: Usability and Interactions with Copilot for Novice Programmers," *ACM Trans. Comput.-Hum. Interact.*, vol. 31, no. 1, pp. 1–31, 2024, doi: 10.1145/3617367.

[23] J. H. Klemmer et al., "Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns," in *Proc. 2024 ACM SIGSAC Conf. Computer and Communications Security (CCS '24)*, Salt Lake City, UT, USA, Oct. 2024, pp. 2726–2740, doi: 10.1145/3658644.3690283.

[24] 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 *Proc. 43rd IEEE Symp. Security and Privacy (SP 2022)*, San Francisco, CA, USA, May 2022, pp. 754–768, doi: 10.1109/SP46214.2022.9833571.

[25] A. Anitha et al., "Impact of Artificial Intelligence on Software Development Processes," *J. Inf. Syst. Eng. Manag.*, vol. 10, no. 25s, 2025.

[26] E. Nurmiati and F. N. Rizqi, "Tinjauan Etika Profesi Teknologi Informasi Terhadap Deepfake Harassment di Media Sosial: Systematic Literature Review," *Jurnal Ilmiah Multidisipin*, vol. 4, no. 4, pp. 368–376, Apr. 2026, doi: 10.60126/jim.v4i4.1563.