BAB II

TINJAUAN PUSTAKA

2.1. Studi Sebelumnya

Penelitian ini merujuk pada enam studi kunci yang menjadi dasar bagi asumsi-asumsi teoretis (*theoretical assumptions*) dalam proses analisis. Sejalan dengan pendekatan *Reflexive Thematic Analysis*, bekal ini berperan sebagai kepekaan awal, bukan sebagai kerangka koding yang kaku, yang mengarahkan perhatian peneliti pada aspek-aspek tertentu dari pengalaman partisipan, namun tetap terbuka terhadap tema-tema baru dari data [13]

Penelitian pertama dilakukan oleh Barke dkk. [11] yang menyajikan analisis *grounded theory* pertama mengenai cara *developer* berinteraksi dengan GitHub Copilot. Melalui pengamatan terhadap 20 partisipan dengan beragam tingkat pengalaman dalam menyelesaikan tugas. Temuan utamanya adalah interaksi bersifat bimodal: dalam mode akselerasi (*acceleration mode*), *developer* sudah mengetahui langkah berikutnya dan memakai Copilot untuk mencapainya lebih cepat. Sementara dalam mode eksplorasi (*exploration mode*), *developer* belum yakin cara melanjutkan sehingga memakai Copilot untuk menjajaki opsi. Penelitian ini menjadi kerangka konseptual penting dalam memahami interaksi manusia dengan asisten berbasis LLM.

Penelitian kedua dilakukan oleh Prather dkk. [14] yang mengeksplorasi usabilitas dan pola interaksi GitHub Copilot bagi mahasiswa pemula yang baru pertama kali menggunakannya. Studi ini mengamati 19 mahasiswa tahun pertama dalam kursus pemrograman pengantar (CS1) yang nyaris tidak memiliki pengalaman sebelumnya dan belum pernah terpapar Copilot. Selain mengonfirmasi pola akselerasi dan eksplorasi temuan Barke dkk., penelitian ini mengidentifikasi dua pola baru khas pemula: *shepherding*, yakni pemula mengetik kode sesuai saran Copilot tetapi akhirnya tidak menerimanya dan *drifting*, yakni pemula menerima kode keliru lalu terjebak *debugging* yang menjauh dari solusi benar, sembari tetap meyakini Copilot selalu menghasilkan

kode benar. Studi ini menjadi rujukan langsung bagi penelitian yang berfokus pada keunikan interaksi pemula.

Penelitian ketiga, juga oleh Prather dkk. [10], mengkaji manfaat dan dampak negatif (*benefits and harms*) penggunaan *Generative AI* bagi pemula. Melalui 21 sesi laboratorium berupa observasi partisipan, wawancara, dan *eye tracking*, studi ini menemukan pembelahan antara mahasiswa yang terakselerasi dan yang kesulitan: bagi yang kesulitan, *Generative AI* memperparah hambatan metakognitif lama bahkan memunculkan hambatan baru, sehingga mereka kerap mengalami disonansi kognitif dan berakhir dengan ilusi kompetensi. Fenomena ini dinamakan sebagai kesenjangan yang melebar (*widening gap*) dan menegaskan bahwa efektivitas alat AI sangat bergantung pada kapasitas regulasi diri dan metakognisi pengguna.

Penelitian keempat, oleh Kazemitabaar dkk. [15], meneliti efek AI *code generators* terhadap pembelajaran pemula melalui eksperimen terkontrol terhadap 69 partisipan dalam kursus pemrograman pengantar. Studi ini membandingkan tiga kondisi pembelajaran: mahasiswa yang menggunakan AI *code generator*, mahasiswa yang menggunakan *code completion* tradisional, dan mahasiswa tanpa bantuan tambahan. Temuan utamanya adalah ketegangan antara manfaat *scaffolding* dan risiko *overreliance*. Di satu sisi, AI menyediakan contoh kode yang dapat dipelajari dan diadaptasi, sehingga mempercepat pemahaman konsep baru. Di sisi lain, akses yang terlalu mudah terhadap solusi instan mendorong perilaku *overreliance*, di mana pemula menjadi terlalu bergantung dan kehilangan kesempatan melatih keterampilan pemecahan masalah mandiri. Studi ini memperkaya pemahaman tentang dampak ganda AI terhadap proses belajar dan menjadi kerangka penting untuk memahami ketegangan serupa yang dialami partisipan dalam penelitian ini.

Penelitian kelima, oleh Lee dkk. [16], menyelidiki dampak AI generatif terhadap kemampuan berpikir kritis melalui survei terhadap 319 pekerja pengetahuan (*knowledge workers*) dari berbagai bidang profesional. Melalui analisis terhadap 936 contoh nyata tugas yang dibantu AI, studi ini menemukan

bahwa kepercayaan yang tinggi pada AI berkaitan dengan berkurangnya usaha berpikir kritis. Fenomena yang disebut sebagai *cognitive offloading*. Temuan ini menunjukkan bahwa pengikisan keterlibatan kognitif bukan hanya terjadi pada pemula, tetapi juga pada profesional, sehingga memperkuat urgensi pemahaman tentang interaksi manusia dan AI. Studi ini relevan dengan argumen tentang pengikisan keterampilan kognitif dalam penelitian ini.

Penelitian keenam, oleh Loksa dkk. [8], menyajikan kerangka teoretis tentang metakognisi dan regulasi diri dalam pendidikan pemrograman. Pemula umumnya dicirikan oleh model mental terbatas dan kemampuan metakognitif yang masih berkembang yaitu kemampuan untuk merencanakan, memantau, dan mengevaluasi proses berpikir sendiri. Akibatnya, mereka kesulitan mengenali ketika mereka tidak memahami suatu konsep, menilai apakah solusi yang ditemukan sudah benar, dan mengatur strategi belajar secara efektif. Dalam konteks penggunaan GitHub Copilot, keterbatasan ini menjadi signifikan karena pemula tidak hanya harus memahami kode yang ditulis, tetapi juga mengevaluasi kualitas saran AI sebuah tugas yang menuntut metakognisi lebih tinggi. Kerangka ini menjadi landasan untuk memahami mengapa pemula rentan terhadap *overreliance* dan ilusi kompetensi ketika berinteraksi dengan Copilot, serta mengapa pola interaksi seperti *shepherding* dan *drifting* [14] muncul secara khas pada kelompok pengguna ini.

Berdasarkan pemaparan di atas, Tabel 2.1. menyajikan perbandingan penelitian terdahulu beserta posisi penelitian ini terhadap masing-masing.

Tabel 2.1. Penelitian Terdahulu

No.	Peneliti	Tujuan	Metode	Hasil Utama	Posisi Penelitian Ini
1.	Barke dkk. (2023)	Menganalisis pola interaksi *developer* dengan Copilot.	*Grounded theory*. Observasi 20 partisipan, empat bahasa.	Pola *bimodal*: mode akselerasi dan mode eksplorasi.	Partisipan beragam pengalaman. Riset ini fokus khusus pemula dan menggali pemaknaan, bukan sekadar pola.
2.	Prather dkk. (2023)	Menyelidiki usabilitas & interaksi pemula dengan Copilot.	Observasi & wawancara 19 mahasiswa CS1 pemula.	Pola baru *shepherding* & *drifting/straying*. Pemula tetap meyakini Copilot selalu benar.	Sesi lab tunggal. Riset ini melacak pengalaman penggunaan nyata sehari-hari plus dimensi sosial.
3.	Prather dkk. (2024)	Mengkaji manfaat & dampak negatif *Generative AI* bagi pemula.	Observasi, wawancara, *eyetracking*, 21 sesi lab.	*Widening gap*: yang kesulitan mengalami hambatan metakognitif baru & ilusi kompetensi.	Berbasis *eyetracking* di lab. Riset ini menggali narasi

No.	Peneliti	Tujuan	Metode	Hasil Utama	Posisi Penelitian Ini
					subjektif & emosi via wawancara dan FGD.
4	Kazemitabaar dkk. (2023)	Meneliti efek *AI code generators* pada pembelajaran pemula	Eksperimen terkontrol pada 69 partisipan	*Scaffolding vs overreliance* dan dampak pada retensi	Memberi kerangka untuk memahami ketegangan manfaat dengan risiko dalam belajar
5	Lee dkk. (2025)	Mengkaji dampak AI pada *critical thinking*	Survei 319 *knowledge workers*	Kepercayaan tinggi pada AI berkorelasi dengan berkurangnya usaha berpikir kritis	Memperkuat argumen tentang pengikisan kognitif pada skala yang lebih luas
6	Loksa dkk. (2022)	Menyajikan kerangka metakognisi & regulasi diri	Tinjauan teoretis	Pemula memiliki metakognisi terbatas. Kesulitan *self regulation*	Landasan teoretis untuk memahami kerentanan pemula terhadap *overreliance*

2.2. Dasar Teori

2.2.1. Large Language Model (LLM)

Large Language Model (LLM) merupakan model kecerdasan buatan yang dibangun di atas arsitektur *Transformer*. Arsitektur ini memperkenalkan mekanisme *self attention* yang memungkinkan model memproses seluruh elemen dalam suatu urutan data secara paralel, serta menangkap ketergantungan antarelemen dalam jarak jauh (*long range dependencies*) secara efisien. Dibandingkan dengan arsitektur sebelumnya seperti *recurrent neural networks* (RNN) yang memproses data secara sekuensial, *Transformer* memiliki jumlah operasi sekuensial yang konstan dan jalur propagasi gradien yang lebih pendek, sehingga lebih unggul dalam hal paralelisasi dan pemodelan konteks jarak jauh. Keunggulan inilah yang menjadikan *Transformer* sebagai fondasi utama bagi pengembangan model bahasa modern berskala besar [17].

LLM dikembangkan melalui proses pre-training berskala besar dengan pendekatan *self supervised*, di mana model dilatih pada korpus teks yang sangat luas tanpa memerlukan pelabelan data secara khusus untuk setiap tugas. Proses ini memungkinkan model mempelajari pola bahasa, hubungan semantik, dan struktur sintaksis secara mendalam. Selain itu, seiring dengan peningkatan skala parameter dan data, LLM menunjukkan kemampuan *emergent* seperti penalaran (*reasoning*), perencanaan, dan generalisasi tugas, serta mampu beradaptasi dengan tugas-tugas baru melalui mekanisme *few shot learning* atau *in context learning* (ICL) tanpa perlu dilakukan penyesuaian parameter (*fine tuning*) secara menyeluruh. Kemampuan ini memungkinkan LLM digunakan untuk berbagai tugas seperti menjawab pertanyaan, merangkum dokumen, menghasilkan teks, dan memahami instruksi yang belum pernah ditemui sebelumnya [18].

2.2.2. GitHub Copilot

GitHub Copilot merupakan asisten pemrograman berbasis kecerdasan buatan yang dikembangkan oleh GitHub yang bekerja sama dengan OpenAI untuk membantu *developer* menghasilkan rekomendasi kode secara kontekstual

selama proses pemrograman. Copilot terintegrasi langsung dengan berbagai *IntegratedDevelopment Environment* (IDE) dan memberikan rekomendasi mulai dari pelengkapan satu baris kode hingga implementasi fungsi secara utuh berdasarkan konteks program yang sedang ditulis.

Penelitian menunjukkan bahwa GitHub Copilot tidak hanya berperan sebagai alat pelengkap kode (*code completion tool*), tetapi juga memengaruhi cara *developer* berinteraksi dengan kode yang dihasilkan. *Developer* memanfaatkan Copilot dalam berbagai pola interaksi, mulai dari menerima rekomendasi secara langsung ketika sudah memiliki gambaran solusi yang jelas, memodifikasi saran yang diberikan, hingga menggunakannya sebagai sarana eksplorasi solusi alternatif ketika belum memiliki gambaran solusi yang pasti [11]

Bagi *developer* pemula, penggunaan GitHub Copilot berpotensi meningkatkan efisiensi penyelesaian tugas dan membantu memahami implementasi kode melalui contoh yang diberikan. Namun demikian, berbagai penelitian juga menunjukkan bahwa rekomendasi yang dihasilkan Copilot tidak selalu benar sehingga pengguna tetap perlu melakukan evaluasi terhadap kualitas, logika, maupun kesesuaian kode dengan kebutuhan sistem [6].

2.2.3. Developer Pemula

Developer pemula dalam penelitian ini mengacu pada klasifikasi status Ferino dkk. [7] yakni mahasiswa aktif rumpun ilmu komputasi yang belum pernah berstatus praktisi profesional pengembang perangkat lunak secara independen di industri. Klasifikasi Ferino dkk. [7] sendiri mencakup dua kelompok populasi *developer* pemula: mahasiswa rumpun ilmu komputasi, dan praktisi awal karier (*early career*) dengan pengalaman profesional maksimal dua tahun. Penelitian ini hanya menyasar kelompok pertama, sehingga status "belum pernah bekerja sebagai praktisi profesional" menjadi kriteria eksklusi mutlak, bukan ambang toleransi hingga dua tahun. Mahasiswa yang pernah bekerja sebagai praktisi pengembang perangkat lunak, meskipun saat ini kembali berstatus mahasiswa aktif, tidak tergolong developer pemula dalam penelitian ini. Status magang (*internship*), tidak termasuk dalam pengecualian ini, karena tetap melekat pada

status mahasiswa aktif, bersifat terstruktur dan disupervisi secara akademik, serta dibatasi waktu, berbeda dari status praktisi profesional yang bekerja secara independen dan permanen.

Pembatasan pada status nonpraktisi ini bukan sekadar kriteria administratif, melainkan berkorelasi dengan sejumlah perbedaan struktural yang relevan terhadap fenomena yang diteliti. Lingkungan industri umumnya menyediakan struktur akuntabilitas atas konsekuensi produksi serta mekanisme kolaborasi dan pengawasan kualitas terlembaga seperti *code review*, *mentoring*, dan *quality assurance* yang secara kolektif mempercepat pematangan metakognisi dan kalibrasi kepercayaan terhadap saran AI. Mahasiswa yang belum berstatus praktisi umumnya mengerjakan tugas pemrograman secara individual tanpa mekanisme pengawasan kualitas eksternal yang setara, sehingga evaluasi terhadap saran Copilot bergantung sepenuhnya pada kapasitas metakognitif dan regulasi diri yang masih berkembang pada kelompok ini. Status keprofesian ini juga bersifat independen dari lama seseorang menggunakan Copilot: mahasiswa yang telah menggunakan Copilot dalam jangka waktu relatif panjang namun belum berstatus praktisi tetap tergolong developer pemula, karena lamanya penggunaan sebuah *tool* tidak mengubah status keprofesian seseorang sebagai *developer*.

Developer pemula merupakan individu yang masih berada pada tahap awal pembentukan kompetensi pemrograman sehingga belum memiliki model mental, strategi pemecahan masalah, dan kemampuan metakognitif yang berkembang sebagaimana developer berpengalaman [8]. Kondisi tersebut menyebabkan proses memahami permasalahan, melakukan debugging, serta mengevaluasi solusi masih memerlukan dukungan belajar yang lebih besar.

Dalam konteks penggunaan GitHub Copilot, *developer* pemula memperoleh manfaat berupa *scaffolding* yang membantu memahami struktur program dan alternatif implementasi kode selama proses belajar. Penyediaan contoh kode secara langsung memungkinkan mahasiswa mempelajari sintaks

maupun pola penyelesaian masalah secara lebih cepat dibandingkan hanya mengandalkan dokumentasi atau mesin pencari [15].

Di sisi lain, beberapa penelitian menemukan bahwa *developer* pemula juga lebih rentan mengalami *overreliance* terhadap rekomendasi AI karena belum memiliki kemampuan yang memadai untuk mengevaluasi kualitas kode secara kritis. Kondisi tersebut dapat menyebabkan munculnya *illusion of competence*, berkurangnya keterlibatan dalam proses berpikir, serta kesulitan memahami kode yang dihasilkan ketika rekomendasi AI tidak sesuai dengan konteks permasalahan [10].

2.2.4. Penelitian Kualitatif

Penelitian kualitatif merupakan pendekatan penelitian yang bertujuan memahami makna suatu fenomena berdasarkan pengalaman, perspektif, dan interpretasi partisipan dalam konteks kehidupan nyata. Pendekatan ini menempatkan peneliti sebagai instrumen utama dalam proses pengumpulan maupun interpretasi data sehingga menghasilkan pemahaman yang mendalam terhadap fenomena yang diteliti [19].

Pendekatan kualitatif sesuai digunakan dalam penelitian ini karena fokus penelitian bukan untuk mengukur efektivitas GitHub Copilot secara statistik, melainkan memahami bagaimana *developer* pemula memaknai pengalaman mereka ketika menggunakan teknologi tersebut dalam proses belajar pemrograman. Melalui pendekatan ini, peneliti dapat mengeksplorasi pengalaman subjektif, persepsi, serta dinamika interaksi antara pengguna dan teknologi yang tidak dapat dijelaskan secara memadai melalui data kuantitatif [13].