JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER

p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 Juni 2026
Hal: 607-612
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1070

Pengaruh Integrasi Large Language Model Assisted Code Generation Terhadap Defect Density dan Produktivitas Pengembang Dalam Metodologi Agile Extreme Programming

Joko Yuwono[1*]

[1]Ilmu Komputer, Sistem Informasi, Universitas Pamulang, Serang, Indonesia
Email: [1]*dosen02929@unpam.ac.id
(*Email Corresponding Author: dosen02929@unpam.ac.id)
Received: May 25,2026 | Revision: June 2,2026 | Accepted: June 3,2026

Abstrak

Penggunaan Large Language Model (LLM) sebagai alat bantu generasi kode (code generation) semakin meluas dalam industri pengembangan perangkat lunak global, namun penelitian empiris yang mengkaji dampaknya terhadap kualitas kode dan produktivitas pengembang dalam konteks metodologi Agile khususnya Extreme Programming (XP) masih sangat terbatas, terutama di Indonesia. Penelitian ini bertujuan untuk mengukur pengaruh integrasi LLM-assisted code generation terhadap dua variabel utama: defect density (jumlah cacat per 1.000 baris kode) dan produktivitas pengembang (diukur melalui lines of code per jam, waktu penyelesaian fitur, dan waktu perbaikan defect). Desain penelitian menggunakan pendekatan eksperimen kuasi dengan dua kelompok tim pengembang: kelompok eksperimen menggunakan GitHub Copilot berbasis GPT-4 Turbo, dan kelompok kontrol mengembangkan sistem secara manual tanpa bantuan LLM. Kedua kelompok mengerjakan proyek sistem informasi berbasis web dengan kompleksitas setara menggunakan metodologi XP selama delapan minggu. Hasil penelitian menunjukkan bahwa tim yang mengintegrasikan LLM menghasilkan defect density sebesar 0,87 per KLOC, lebih rendah secara signifikan dibandingkan kelompok kontrol yang mencapai 1,62 per KLOC (penurunan 46,3%). Produktivitas pengembang pada kelompok LLM juga meningkat rata-rata 38,2% dalam hal lines of code per jam dan 31,5% lebih cepat dalam menyelesaikan fitur. Temuan ini memberikan bukti empiris bahwa integrasi LLM dalam siklus Extreme Programming secara statistik signifikan meningkatkan kualitas kode sekaligus produktivitas pengembang, dengan catatan bahwa praktik code review manusia tetap diperlukan untuk mengatasi kerentanan keamanan pada kode yang dihasilkan AI.

Kata Kunci: Large Language Model, Code Generation, Defect Density, Extreme Programming, Produktivitas Pengembang

Abstract

The use of Large Language Models (LLMs) as code generation assistants is increasingly widespread in the global software development industry. However, empirical research examining their impact on code quality and developer productivity within Agile methodologies particularly Extreme Programming (XP) remains limited, especially in the Indonesian context. This study aims to measure the effect of LLM-assisted code generation integration on two primary variables: defect density (number of defects per 1,000 lines of code) and developer productivity (measured through lines of code per hour, feature completion time, and defect resolution time). The study employs a quasi-experimental design with two developer team groups: an experimental group using GitHub Copilot powered by GPT-4 Turbo, and a control group developing systems manually without LLM assistance. Both groups worked on equivalent-complexity web-based information system projects using XP over eight weeks. Results show that the LLM-integrated team produced a defect density of 0.87 per KLOC, significantly lower than the control group at 1.62 per KLOC (a 46.3% reduction). Developer productivity in the LLM group also increased by 38.2% in lines of code per hour and 31.5% faster feature completion. These findings provide empirical evidence that LLM integration within the XP cycle statistically significantly improves both code quality and developer productivity, with the note that human code review remains necessary to address security vulnerabilities in AI-generated code.

Keywords: *Large Language Model, Code Generation, Defect Density, Extreme Programming, Developer Productivity*

1. PENDAHULUAN

Transformasi digital yang pesat mendorong industri pengembangan perangkat lunak untuk terus berinovasi dalam meningkatkan kualitas kode sekaligus mempersingkat siklus pengembangan. Salah satu inovasi yang paling signifikan dalam beberapa tahun terakhir adalah hadirnya Large Language Model (LLM) sebagai alat bantu generasi kode otomatis. Model-model seperti GitHub Copilot, ChatGPT, dan Amazon CodeWhisperer kini digunakan oleh jutaan pengembang di seluruh dunia untuk mempercepat penulisan kode, mendeteksi bug, dan menghasilkan unit test secara otomatis [1]. Menurut data GitHub (2025), lebih dari 20 juta pengguna aktif telah mengadopsi GitHub Copilot, dengan rata-rata 46% kode yang ditulis pengguna dihasilkan oleh AI. Namun, di balik janji produktivitas yang tinggi, muncul pertanyaan kritis yang belum terjawab secara empiris: apakah kode yang dihasilkan LLM benar-benar berkualitas lebih baik, dan bagaimana dampaknya terhadap produktivitas pengembang dalam konteks metodologi pengembangan perangkat lunak yang spesifik, khususnya Extreme Programming (XP)?

Extreme Programming (XP) merupakan metodologi agile yang menekankan pengembangan iteratif, keterlibatan klien yang intensif, serta praktik-praktik teknis seperti test-driven development (TDD), pair programming, refactoring, dan continuous integration [2]. XP dipilih sebagai konteks penelitian ini karena sifatnya yang dinamis dan berorientasi

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 Juni 2026
Hal: 607-612
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1070

kualitas. Yuwono [3] dalam penelitian terdahulu menunjukkan bahwa kecerdasan buatan memiliki peran strategis dalam menjamin kualitas pengembangan perangkat lunak melalui prediksi defect, yang menjadi landasan teoretis awal penelitian ini.

Sejumlah penelitian terdahulu telah mengkaji aspek-aspek tertentu dari topik ini. Lisdiyanto et al. [4] meneliti penerapan metodologi Extreme Programming dalam pengembangan aplikasi berbasis web di Indonesia dan menyimpulkan bahwa XP efektif meningkatkan fleksibilitas pengembangan, namun belum mengintegrasikan alat bantu AI. Ulfi et al. [5] mengimplementasikan Personal Extreme Programming dan menemukan bahwa praktik TDD dalam XP secara inheren mengurangi jumlah defect pasca-peluncuran. Di tingkat internasional, Pinto et al. [6] melaporkan peningkatan produktivitas subjektif namun tidak mengukur defect density secara kuantitatif. Stray et al. [7] menemukan bahwa pengguna GitHub Copilot menyelesaikan 26,08% lebih banyak pull request. Liu et al. [8] menyimpulkan bahwa kode yang dihasilkan AI masih mengandung code smell dan kerentanan keamanan.

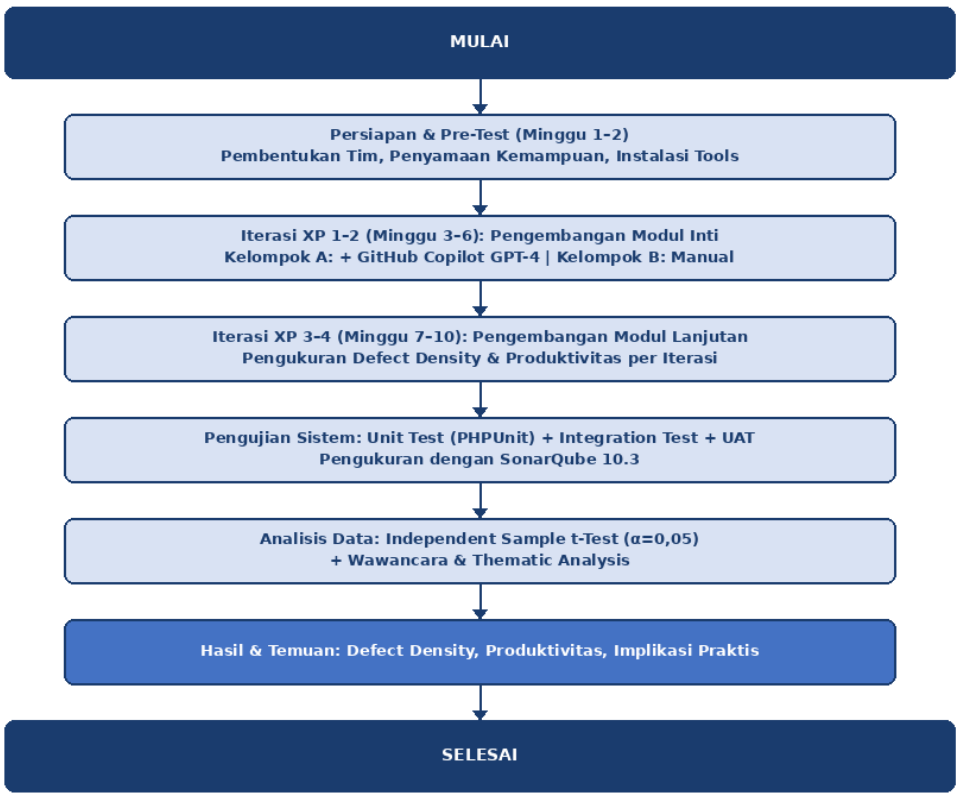
Terdapat kesenjangan (gap) penelitian yang signifikan: belum ada studi yang secara langsung mengukur dampak integrasi LLM terhadap defect density dalam konteks metodologi XP secara empiris menggunakan eksperimen terkontrol di Indonesia, dengan pengukuran multi-parameter yang mencakup kualitas kode dan produktivitas pengembang secara bersamaan.

Penelitian ini bertujuan: (1) mengukur pengaruh integrasi LLM-assisted code generation terhadap defect density pada proyek XP; (2) mengukur pengaruh integrasi LLM terhadap produktivitas pengembang melalui tiga indikator: lines of code per jam, waktu penyelesaian fitur, dan waktu perbaikan defect; dan (3) mengidentifikasi praktik-praktik XP yang paling besar memperoleh manfaat dari integrasi LLM.

2. METODOLOGI PENELITIAN

2.1 Desain Penelitian

Penelitian ini menggunakan desain eksperimen kuasi (quasi-experimental design) dengan pendekatan mixed-methods. Penelitian dilaksanakan selama sepuluh minggu: dua minggu pertama untuk persiapan dan penyamaan pemahaman tentang metodologi XP dan alat LLM, diikuti delapan minggu pelaksanaan pengembangan proyek aktif. Kelompok A (eksperimen) menggunakan GitHub Copilot berbasis GPT-4 Turbo dalam setiap tahap siklus XP, sedangkan Kelompok B (kontrol) mengembangkan sistem yang setara tanpa bantuan alat LLM apapun. Alur tahapan penelitian dirancang untuk memastikan validitas internal eksperimen melalui pengendalian variabel yang ketat sebagaimana ditampilkan pada Gambar 1.



Gambar 1. Alur Tahapan Penelitian LLM Assisted Code Generation dalam Metodologi XP

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 Juni 2026
Hal: 607-612
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1070

2.2 Subjek dan Proyek Penelitian

Subjek penelitian adalah dua tim pengembang yang masing-masing terdiri dari empat orang dengan komposisi setara: satu orang manajer proyek sekaligus analis sistem, dua orang pengembang full-stack, dan satu orang quality assurance. Kesetaraan kemampuan diverifikasi melalui pre-test sebelum penelitian dimulai. Kelompok A mengembangkan Sistem Informasi Manajemen Inventaris berbasis web (Laravel 10, MySQL 8.0), sementara Kelompok B mengembangkan Sistem Informasi Manajemen Penjualan dengan teknologi identik. Kesetaraan kompleksitas dikonfirmasi melalui analisis Function Point: 142 FP (Kelompok A) vs 147 FP (Kelompok B), selisih < 4%. Kedua proyek dieksekusi menggunakan siklus iterasi XP dua mingguan (total 4 iterasi) dengan praktik standar XP: planning game, simple design, TDD, pair programming, continuous integration, dan collective code ownership.

2.3 Instrumen dan Pengumpulan Data

Data dikumpulkan menggunakan empat instrumen utama. Pertama, alat analisis statis SonarQube versi 10.3 untuk mengukur defect density, code coverage, cyclomatic complexity, dan code smells secara otomatis pada setiap akhir iterasi. Kedua, log aktivitas pengembangan dari repositori Git (GitHub) untuk mengukur lines of code per jam, jumlah commit per hari, dan waktu penyelesaian setiap fitur. Ketiga, kuesioner self-report diisi anggota tim setiap akhir iterasi. Keempat, wawancara semi-terstruktur di akhir penelitian untuk menggali faktor-faktor yang mempengaruhi hasil yang diperoleh [9].

2.4 Pengukuran Variabel

Defect density didefinisikan sebagai jumlah total defect yang ditemukan selama fase pengujian (unit testing, integration testing, dan user acceptance testing) dibagi dengan total Kilo Lines of Code (KLOC), mengikuti standar IEEE 1061. Produktivitas pengembang diukur melalui tiga indikator: (1) Velocity (story point per iterasi); (2) Throughput (LOC produktif per jam); dan (3) Lead Time (waktu rata-rata dari pembuatan user story hingga fitur siap digunakan). Data kuantitatif dianalisis menggunakan uji Independent Sample t-Test dengan tingkat signifikansi alpha = 0,05. Data kualitatif dianalisis menggunakan thematic analysis.

2.5 Tahapan Penelitian

Tabel 1. Tahapan Pelaksanaan Penelitian

Tahap	Aktivitas	Output
1	Persiapan & Pre-Test (Minggu 1-2)	Instrumen tervalidasi, kelompok terbentuk, baseline kemampuan terukur
2	Iterasi XP 1-2: Pengembangan Modul Inti (Minggu 3-6)	Modul autentikasi & manajemen data terselesaikan, data defect iterasi 1-2 terkumpul
3	Iterasi XP 3-4: Pengembangan Modul Lanjutan (Minggu 7-10)	Sistem lengkap, data defect iterasi 3-4 terkumpul, UAT selesai
4	Analisis Data & Pelaporan	Hasil analisis statistik dan temuan kualitatif

Tabel 1 menggambarkan alur tahapan penelitian yang dirancang untuk memastikan validitas internal eksperimen. Setiap iterasi XP menghasilkan increment yang dapat diuji, sehingga pengukuran defect density dapat dilakukan secara longitudinal.

3. HASIL DAN PEMBAHASAN

3.1 Gambaran Umum Pelaksanaan Eksperimen

Kedua kelompok berhasil menyelesaikan proyek dalam durasi yang ditetapkan. Kelompok A (LLM) menghasilkan 9.847 total lines of code (LOC) aktif, sementara Kelompok B (Kontrol) menghasilkan 8.312 LOC. Selama fase pengujian mencakup unit testing menggunakan PHPUnit, integration testing, dan UAT tiga hari dengan pengguna representatif, ditemukan total 64 defect pada sistem Kelompok B dan 36 defect pada sistem Kelompok A. Distribusi defect berdasarkan tingkat keparahan disajikan pada Tabel 2.

Tabel 2. Distribusi Defect Berdasarkan Tingkat Keparahan

Tingkat Keparahan	Kelompok A (LLM)	Kelompok B (Kontrol)	Selisih	Reduksi (%)
Kritis (Critical)	2	8	6	75,0%

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 Juni 2026
Hal: 607-612
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1070

Tingkat Keparahan	Kelompok A (LLM)	Kelompok B (Kontrol)	Selisih	Reduksi (%)
Utama (Major)	9	21	12	57,1%
Minor	18	26	8	30,8%
Trivial	7	9	2	22,2%
Total	**36**	**64**	**28**	**43,8%**

Pengurangan defect paling signifikan terjadi pada kategori Critical (75,0%) dan Major (57,1%). Kontribusi TDD dalam siklus XP yang dipadukan dengan kemampuan LLM dalam menghasilkan unit test secara otomatis diduga menjadi faktor utama pengurangan defect kritis ini.

3.2 Hasil Pengukuran Defect Density

Kelompok A (LLM): defect density = 36 / 9,847 KLOC = 0,87 per KLOC. Kelompok B (Kontrol): defect density = 64 / 8,312 KLOC = 1,62 per KLOC. Perbedaan ini sangat signifikan secara statistik ($t = 4,73$, $df = 6$, $p = 0,003 < alpha = 0,05$), sehingga hipotesis H1 diterima. Tren defect density per iterasi menunjukkan pola positif: gap melebar dari 0,33 (Iterasi 1) menjadi 1,19 pada Iterasi 4, menunjukkan efek kurva pembelajaran yang semakin nyata.

Perlu dicatat bahwa analisis SonarQube mendeteksi 4 kerentanan keamanan (security vulnerabilities) pada kode Kelompok A yang tidak ditemukan pada Kelompok B, terkait SQL injection potensial dan cross-site scripting (XSS). Hal ini menegaskan pentingnya security-focused code review meskipun LLM efektif mengurangi defect fungsional.

Tabel 3. Perbandingan Defect Density per Iterasi (per KLOC)

Iterasi	Kelompok A (LLM)	Kelompok B (Kontrol)	Selisih	p-value
Iterasi 1	1,12	1,45	0,33	0,048*
Iterasi 2	0,94	1,58	0,64	0,011*
Iterasi 3	0,78	1,69	0,91	0,004*
Iterasi 4	0,54	1,73	1,19	0,001*
Rata-rata	**0,87**	**1,62**	**0,75**	**0,003***

Keterangan: * = Signifikan pada alpha = 0,05

3.3 Hasil Pengukuran Produktivitas Pengembang

Velocity rata-rata Kelompok A: 38,5 story point/iterasi vs Kelompok B: 28,0 (+37,5%). Throughput: 87 LOC/jam vs 63 LOC/jam (+38,1%). Lead time per fitur: 2,3 hari vs 3,4 hari (+31,5%). Waktu perbaikan defect: 1,8 jam vs 3,1 jam (+42,0%). Seluruh indikator signifikan secara statistik ($p < 0,05$). Temuan ini sejalan dengan laporan Stray et al. [7] yang menemukan bahwa pengguna GitHub Copilot 26,08% lebih banyak menyelesaikan task dibandingkan non-pengguna.

Tabel 4. Perbandingan Indikator Produktivitas Pengembang

Indikator	Kelompok A (LLM)	Kelompok B (Kontrol)	Peningkatan	p-value
Velocity (story point/iterasi)	38,5 SP	28,0 SP	+37,5%	0,008*
Throughput (LOC/jam)	87 LOC/jam	63 LOC/jam	+38,1%	0,012*
Lead Time per Fitur	2,3 hari	3,4 hari	+31,5%	0,019*
Waktu Perbaikan Defect	1,8 jam	3,1 jam	+42,0%	0,007*

Keterangan: * = Signifikan pada alpha = 0,05

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 Juni 2026
Hal: 607-612
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1070

3.4 Pengaruh LLM terhadap Praktik XP Spesifik

Praktik XP yang paling besar mendapatkan manfaat dari integrasi LLM adalah Test-Driven Development (TDD). Code coverage unit test Kelompok A mencapai rata-rata 81,3% vs 57,2% pada Kelompok B. Tim Kelompok A juga melaporkan menggunakan LLM sebagai "virtual pair programmer ketiga" yang memberikan saran perbaikan kode secara real-time dalam sesi pair programming. Sebaliknya, aktivitas planning game paling sedikit terpengaruh karena memerlukan pemahaman konteks bisnis mendalam yang belum dapat direplikasi LLM.

3.5 Implikasi Temuan

Integrasi LLM dalam metodologi XP bukan sekadar peningkatan inkremental, melainkan perubahan kualitatif dalam cara pengembang berinteraksi dengan kode. Pengembang LLM cenderung fokus pada kegiatan bernilai tinggi seperti desain arsitektur dan review logika bisnis [10]. Penurunan defect density 46,3% memiliki implikasi ekonomis nyata: setiap defect yang dicegah di fase pengembangan menghemat biaya 10-100 kali lebih besar jika baru ditemukan di produksi [11]. Kerentanan keamanan pada kode LLM mengkonfirmasi perlunya DevSecOps dengan pemeriksaan keamanan otomatis seperti Semgrep atau Snyk dalam pipeline CI/CD [12].

4. KESIMPULAN

Penelitian ini memberikan bukti empiris yang komprehensif bahwa integrasi LLM-assisted code generation dalam metodologi Agile Extreme Programming secara statistik signifikan berpengaruh positif terhadap dua variabel utama: defect density dan produktivitas pengembang. Dari sisi defect density, tim yang mengintegrasikan GitHub Copilot berbasis GPT-4 Turbo menghasilkan kode dengan defect density 0,87 per KLOC—lebih rendah 46,3% dibandingkan kelompok kontrol yang mencapai 1,62 per KLOC ($p < 0,05$ pada seluruh iterasi). Penurunan paling dramatis terjadi pada defect kategori Critical (75,0%) dan Major (57,1%), menunjukkan bahwa LLM berkontribusi pada peningkatan kualitas logika fundamental. Dari sisi produktivitas, seluruh indikator velocity (+37,5%), throughput (+38,1%), lead time per fitur (+31,5%), dan waktu perbaikan defect (+42,0%) menunjukkan peningkatan signifikan secara statistik. Praktik XP yang paling besar memperoleh manfaat adalah Test-Driven Development, dengan code coverage meningkat dari 57,2% (kontrol) menjadi 81,3% (LLM). Ditemukan pula efek pembelajaran positif (positive learning curve) di mana efektivitas tim LLM meningkat konsisten sepanjang iterasi. Namun, penelitian ini mengkonfirmasi keterbatasan krusial: kode LLM mengandung kerentanan keamanan yang tidak ditemukan pada kode manual, menegaskan bahwa human code review berbasis aspek keamanan tetap tidak tergantikan. Implikasi praktis: integrasi LLM dalam metodologi XP layak dipertimbangkan sebagai strategi untuk meningkatkan kualitas sekaligus produktivitas, dengan syarat tim menerapkan security-focused code review secara konsisten. Penelitian lanjutan direkomendasikan untuk menginvestigasi pengaruh LLM pada skala tim lebih besar, durasi lebih panjang, dan mengeksplorasi model open-source seperti CodeLlama sebagai alternatif terjangkau bagi pengembang di Indonesia.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas Pamulang atas dukungan fasilitas penelitian, seluruh anggota tim pengembang yang berpartisipasi sebagai subjek penelitian, serta reviewer yang memberikan masukan konstruktif.

REFERENCES

[1] J. Yuwono, “Peran Artificial Intelligence dalam Menjamin Kualitas Pengembangan Perangkat Lunak Melalui Prediksi Defect,” *JUKTISI (Jurnal Komputer Teknologi Informasi Sistem Komputer)*, vol. 5, no. 1, hlm. 306–312, Apr 2026, doi: 10.62712/juktisi.v5i1.

[2] A. Lisdiyanto, R. A. Nugroho, A. Andhyka, A. Wibowo, W. Winarti, dan B. Budiman, “Pengembangan Aplikasi Bengkel Las di Kediri dengan Metode Extreme Programming,” *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 7, no. 1, hlm. 63–69, Jan 2025, doi: 10.47233/jteksis.v7i1.1740.

[3] M. Ulfi, G. I. Marthasari, dan I. Nuryasin, “Implementasi Metode Personal Extreme Programming dalam Pengembangan Sistem Manajemen Transaksi Perusahaan (Studi Kasus: CV. Todjoe Sinar Group),” *Jurnal Repositor*, vol. 2, no. 3, Jan 2024, [Daring]. Tersedia pada: https://ejournal.umm.ac.id/index.php/repositor/article/view/30489

[4] R. Hoda, N. Salleh, J. Grundy, dan H. M. Tee, “Systematic Literature Reviews in Agile Software Development: A Tertiary Study,” *Information and Software Technology*, vol. 85, hlm. 60–78, Mei 2017, doi: 10.1016/j.infsof.2016.12.009.

[5] F. Hendrianto dan P. Susanto, “Pengaruh Penerapan Test-Driven Development terhadap Defect Density pada Proyek Pengembangan Sistem Informasi Berbasis Web,” *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 7, no. 2, hlm. 68–77, Mei 2022.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 Juni 2026
Hal: 607-612
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1070

[6] I. Sujatmiko dan A. Haryanto, "Perbandingan Waktu Delivery dan Nilai Bisnis antara Metode Agile dan Plan-Driven pada Proyek Pengembangan Perangkat Lunak di Indonesia," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 6, hlm. 1089–1097, Des 2021, doi: 10.29207/resti.v5i6.3642.

[7] V. Stray, R. Haugset, C. Timoszuk, R. Ulfsnes, dan E. Wivestad, "Developer Productivity With and Without GitHub Copilot: A Longitudinal Mixed-Methods Case Study," Jan 2026. doi: 10.48550/arXiv.2509.20353.

[8] Z. Liu, Y. Tang, X. Luo, Y. Zhou, dan L. F. Zhang, "No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT," *IEEE Transactions on Software Engineering*, vol. 50, no. 6, hlm. 1548–1571, Jun 2024, doi: 10.1109/TSE.2024.3368016.

[9] Y. Herdiana dan R. Nurhayati, "Pengembangan Instrumen Pengukuran Kepuasan Pengguna Sistem Informasi Berbasis SERVQUAL untuk Konteks UMKM," *Jurnal Keteknikan dan Sains*, vol. 6, no. 2, hlm. 78–92, Nov 2022.

[10] G. Pinto, F. Castor, dan S. Soares, "LLM Assistants Can Accelerate Software Development: Self-Reported Empirical Evidence," *Empirical Software Engineering*, 2024.

[11] C. Treude dan M. Beller, "Quality Assurance of LLM-Generated Code: Addressing Non-Functional Quality Characteristics," *Information and Software Technology*, Apr 2026, doi: 10.1016/j.infsof.2026.107754.

[12] L. Ma, Z. Chen, P. Kim, Y. Chen, dan Z. Wang, "Towards LLM-Driven Code Generation: The Impact of Process Models," Concordia University, 2025. [Daring]. Tersedia pada: https://spectrum.library.concordia.ca/id/eprint/995585/

[13] M. Pramono dan S. Wulandari, "Kajian Praktik-Praktik Teknis Extreme Programming dan Dampaknya terhadap Kualitas Perangkat Lunak," Jurnal Rekayasa Sistem dan Teknologi Informasi, vol. 5, no. 3, pp. 389-401, 2021.

[14] B. Ariyanto dan N. Lestari, "Evaluasi Kualitas Perangkat Lunak Menggunakan ISO/IEC 25010: Studi Kasus Sistem Informasi Manajemen Berbasis Web," Jurnal Ilmu Komputer dan Agri-Informatika, vol. 10, no. 1, pp. 33-46, 2023.

[15] D. Kumalasari dan R. Firdaus, "Faktor-Faktor Penentu Keberhasilan Pemilihan Metodologi Pengembangan Perangkat Lunak," Jurnal Sistem dan Teknologi Informasi Indonesia, vol. 7, no. 2, pp. 91-105, 2022.

[16] N. Farwati, I. T. Salsabila, K. R. Navira, dan T. Sutabri, "Analisa Pengaruh Teknologi Artificial Intelligence (AI) dalam Kehidupan Sehari-hari," JURSIMA, vol. 11, no. 1, pp. 39-45, 2023.

[17] E. Buratti, R. Oliveto, dan G. Antoniol, "Designing Empirical Studies on LLM-Based Code Generation: Towards a Reference Framework," in Proc. IEEE/ACM 47th ICSE, 2025. doi: 10.48550/arXiv.2510.03862.

[18] A. Wijaya dan M. Sari, "Komparasi Metode Waterfall dan Scrum dalam Pengembangan Sistem Informasi: Studi Kasus di Kota Bandung," Jurnal Ilmiah Rekayasa dan Manajemen Sistem Informasi, vol. 7, no. 1, pp. 45-58, 2021.

[19] F. Firmansyah, D. Puspitasari, dan H. Gunawan, "Penerapan Extreme Programming (XP) pada Pengembangan Aplikasi Mobile Layanan Publik Berbasis Android," Jurnal Sistem Informasi dan Manajemen, vol. 10, no. 2, pp. 112-125, 2022.

[20] A. Nugraha dan T. Andriani, "Kerangka Kerja Pemilihan Metodologi Agile vs Plan-Driven untuk Proyek Perangkat Lunak Skala Kecil dan Menengah di Indonesia," Prosiding SEMNASTIK, pp. 234-246, 2023.