Majalah Ilmiah Teknologi Elektro, Vol.25, No.1, Jan-Juni 2026
DOI: https://doi.org/10.24843/MITE.2026.v25i01.P13

Pemanfaatan *Local Large Language Model* dalam Software *Effort Estimation* Menggunakan Metode LOOCV

Diyan Sueka Buana Putra[1], Dewa Made Wiharta[2*], Widyadi Setiawan[3], Ida Bagus Gede Manuaba[4]
[Submission: 11-06-2026, Accepted: 30-06-2026]

Abstract*— Software Effort Estimation (SEE) in the early stages of a project is generally confronted with minimal technical specifications and high ambiguity in requirement documents. Large Language Models (LLMs) offer the potential to automate information extraction from unstructured natural language text. This exploratory study evaluates the standalone performance of a local LLM (Phi-3 Mini architecture) in predicting man-days effort using a within-company historical dataset from a local digital agency. The experiment was conducted using the Leave-One-Out Cross-Validation (LOOCV) method to ensure objective evaluation within a limited data population. Model optimization utilized Chain-of-Thought (CoT) prompt engineering combined with a methodological strategy of few-shot selection based on boundary distribution (Min, Median, Max) to serve as cognitive anchors. Qualitative analysis proves that the LLM possesses high semantic intelligence in evaluating relative project complexity. However, quantitative evaluation using MMRE, PRED(25), and Coefficient of Determination (R-squared) indicates fundamental limitations in standalone mathematical regression. The model achieved an MMRE of 0.6963 and a PRED(25) of 0.25, performing below the Mean Baseline (MMRE 0.44) with a negative R-squared of -2.038. These results confirm that while LLMs are proficient in understanding context, they struggle with precise numerical interpolation. Consequently, this study recommends a hybrid estimation framework for future research, positioning the LLM purely as a qualitative feature extractor for conventional Machine Learning algorithms to produce robust and precise estimation accuracy.

***Intisari*— Estimasi Upaya Perangkat Lunak pada fase awal proyek umumnya dihadapkan pada minimnya spesifikasi teknis dan tingginya ambiguitas dokumen kebutuhan. *Large Language Model* (LLM) menawarkan potensi automasi ekstraksi informasi dari teks bahasa alami yang tidak terstruktur. Penelitian eksploratif ini mengevaluasi kinerja local LLM (arsitektur Phi-3 Mini) secara mandiri dalam memprediksi estimasi hari-kerja (*man-days*) menggunakan dataset internal dari sebuah agensi digital lokal. Pengujian dilakukan menggunakan metode *Leave-One-Out Cross-Validation* (LOOCV) untuk menjamin objektivitas evaluasi pada populasi data yang terbatas. Optimasi model diimplementasikan melalui rekayasa prompt *Chain-of-Thought* (CoT) yang dipadukan dengan strategi metodologis *few-shot* berbasis representasi distribusi batas (*Min, Median, Max*) sebagai jangkar kognitif. Hasil analisis kualitatif membuktikan bahwa LLM memiliki kecerdasan semantik yang tinggi dalam mengevaluasi kompleksitas relatif proyek. Namun, evaluasi kuantitatif menggunakan metrik MMRE, PRED(25), dan *Koefisien Determinasi (R-squared)* menunjukkan adanya keterbatasan fundamental model dalam tugas regresi matematis secara mandiri. Model menghasilkan nilai MMRE sebesar 0,6963 dan PRED(25) sebesar 0,25, di mana performa tersebut masih di bawah Mean Baseline (MMRE 0,44) dengan nilai R-squared negatif sebesar -2,038. Sebagai solusi, penelitian ini merekomendasikan pengembangan kerangka kerja estimasi hibrida pada penelitian mendatang. Melalui pendekatan hibrida ini, kemampuan analisis teks dari LLM diposisikan murni sebagai pengekstraksi fitur kualitatif, yang kemudian diumpankan ke dalam algoritma *Machine Learning* konvensional untuk menghasilkan akurasi prediksi yang presisi dan tangguh.**

***Kata Kunci*— Software Effort Estimation; Large Language Model; Phi-3;* Akurasi Estimasi; LOOCV.**

I. PENDAHULUAN

Estimasi upaya pengembangan perangkat lunak (*Software Effort Estimation* atau SEE) merupakan tahapan penting dalam manajemen proyek yang menentukan alokasi sumber daya, penjadwalan, dan anggaran. Tantangan utama yang persisten di industri perangkat lunak adalah tingginya ketergantungan pada penilaian pakar (*expert judgment*) yang rentan terhadap bias kognitif dan subjektivitas, sehingga sering kali menghasilkan estimasi yang tidak konsisten [3].

Pendekatan berbasis data (*data-driven*) menggunakan algoritma *Machine Learning* telah banyak diadopsi untuk meningkatkan objektivitas estimasi. Meskipun model statistik numerik unggul dalam mengenali pola historis, pendekatan ini memiliki keterbatasan mendasar berupa ketidakmampuan memahami nuansa kualitatif dan konteks dari deskripsi spesifikasi kebutuhan proyek (*requirements*) yang disajikan dalam bahasa alami [4],[6],[18].

Perkembangan pesat kecerdasan buatan generatif, khususnya *Large Language Model* (LLM), membuka peluang baru untuk mengatasi kelemahan pemahaman konteks tersebut. LLM memiliki kapabilitas penalaran kontekstual yang mampu memproses teks spesifikasi perangkat lunak yang tidak terstruktur secara mendalam [7]. Beberapa studi terkini menunjukkan bahwa antarmuka berbasis LLM dapat memperkaya analisis kebutuhan proyek [2], meskipun implementasinya masih rentan terhadap fenomena halusinasi numerik jika tidak dievaluasi dengan ketat [5].

Mayoritas penelitian sebelumnya berfokus pada penggunaan LLM komersial berskala masif yang membutuhkan koneksi internet, sehingga memunculkan kekhawatiran terkait privasi dan keamanan data historis perusahaan. Penggunaan *local* LLM berskala kecil (*Small Language Models*) seperti varian Phi-3 menawarkan alternatif komputasi yang aman untuk memproses data spesifik di lingkungan internal secara luring [1]. Meskipun demikian,

[1]*Mahasiswa, Magister Teknik Elektro Universitas Udayana, Denpasar, 80237 (e-mail: diyansueka77@gmail.com)*
[2,3,4]*Jurusan Teknik Elektro Fakultas Teknik Universitas Udayana, Jln. Jalan Kampus Bukit Jimbaran 80361 Indonesia (wiharta@unud.ac.id., widyadi@unud.ac.id., ibgmanuaba@unud.ac.id) Corresponding : wiharta@unud.ac.id)*

p-ISSN:1693 – 2951; e-ISSN: 2503-2372

ketangguhan *local* LLM dalam menghasilkan estimasi angka secara mandiri (*standalone*) belum teruji secara ekstensif pada studi kasus skala industri riil.

Penelitian ini bertujuan untuk mengevaluasi kemampuan penalaran dan akurasi *local* LLM secara mandiri dalam melakukan estimasi upaya perangkat lunak, menggunakan dataset historis dari *digital agency* lokal. Pendekatan ini murni memanfaatkan teknik *few-shot prompting* pada LLM lokal untuk memproses narasi deskripsi proyek, tanpa melibatkan algoritma regresi konvensional. Untuk memastikan validitas dan objektivitas evaluasi pada dataset riil yang jumlahnya terbatas, penelitian ini mengimplementasikan metode validasi silang *Leave-One-Out Cross-Validation* (LOOCV). Kinerja prediksi model akan diukur secara spesifik menggunakan metrik standar industri, yaitu *Mean Magnitude of Relative Error* (MMRE) untuk mengevaluasi rata-rata galat presisi prediksi, serta *R-squared* [R^2] untuk mengukur tingkat kecocokan prediksi LLM terhadap variabilitas estimasi upaya yang sebenarnya (aktual).

II. STUDI PUSTAKA

A. Evolusi Pendekatan Software Effort Estimation (SEE)

Secara historis, praktik *Software Effort Estimation* (SEE) sangat bergantung pada metode penilaian pakar (*expert judgment*) dan model algoritmik parametrik seperti COCOMO yang diperkenalkan pada penelitian [14]. Namun, pada penelitian [3] mencatat bahwa estimasi berbasis pakar sering kali rentan terhadap bias kognitif dan inkonsistensi. Sebagai solusinya, pendekatan berbasis *Machine Learning* (ML) mulai mendominasi literatur SEE. Berbagai studi sistematis [4], [6],[16], membuktikan bahwa teknik ML mampu memberikan tingkat akurasi pemetaan data historis yang lebih adaptif dibandingkan model matematis kaku. Meskipun demikian, model ML konvensional memiliki kelemahan mendasar: algoritma tersebut membutuhkan ekstraksi fitur numerik atau kategorikal yang terstruktur secara ketat, sementara pada fase awal proyek, spesifikasi kebutuhan dari klien umumnya masih berupa teks bahasa alami yang ambigu dan tidak terstruktur.

B. Pemanfaatan Large Language Model dalam Rekayasa Perangkat Lunak

Kemajuan pesat dalam *Natural Language Processing* (NLP) melahirkan *Large Language Models* (LLM) yang mampu memahami konteks semantik dari teks mentah. Survei komprehensif pada [12], [13] menunjukkan bahwa LLM telah berhasil diaplikasikan secara luas dalam berbagai tugas rekayasa perangkat lunak, mulai dari pembangkitan kode hingga analisis kebutuhan. Memasuki *state-of-the-art* terkini, literatur mulai mengeksplorasi potensi LLM langsung untuk tugas SEE. Penelitian pada [2] serta tinjauan pemetaan sistematis dalam [5] menginvestigasi penggunaan antarmuka cerdas berbasis LLM untuk menaksir estimasi proyek pada tahap awal. Senada dengan hal tersebut, studi pada [7] mengajukan kerangka kerja teoretis untuk meningkatkan SEE dengan memanfaatkan kemampuan analitik AI generatif. Sebagian besar penelitian ini menyimpulkan bahwa LLM memiliki potensi besar dalam membaca spesifikasi proyek secara instan.

C. Keterbatasan Regresi Numerik LLM dan Strategi Prompting

Meskipun LLM sangat handal dalam tugas berbasis bahasa, arsitektur utamanya adalah mesin probabilitas teks (*next-token predictor*), bukan kalkulator matematis. Hal ini menyebabkan LLM sering mengalami halusinasi atau bias saat dihadapkan pada tugas regresi kontinu numerik secara pasti. Untuk memitigasi hal ini, teknik *Prompt Engineering* tingkat lanjut mulai diterapkan. Penelitian [10] membuktikan bahwa memberikan beberapa contoh (*Few-Shot Prompting*) dapat secara drastis meningkatkan akurasi LLM dalam mengenali pola tugas spesifik tanpa perlu melakukan *fine-tuning*. Lebih lanjut, [9] memperkenalkan *Chain-of-Thought* (CoT), sebuah metode yang memaksa model untuk menjabarkan langkah-langkah penalaran sebelum memberikan jawaban akhir [19]. Penggunaan kombinasi *Few-Shot* dan CoT terbukti efektif memandu logika deduktif, meskipun evaluasi ketat mengenai kemampuannya dalam regresi *effort* berskala rasio (*man-days*) masih terbatas.

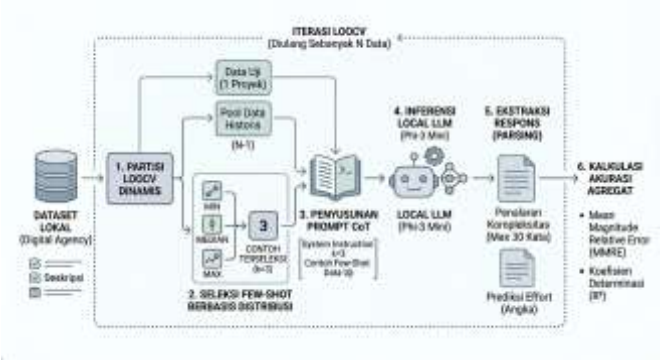
D. State-of-the-Art dan Posisi Penelitian

Penelitian-penelitian SEE berbasis LLM yang telah terbit saat ini [2], [5], dan [7] mayoritas berfokus pada penggunaan model bahasa komersial berbasis *cloud* dan melakukan pengujian pada dataset publik berskala besar (*cross-company data*). Selain itu, literatur tersebut sering berasumsi bahwa LLM dapat langsung memberikan angka estimasi yang presisi jika diberikan *prompt* yang baik. Menanggapi *state-of-the-art* tersebut, penelitian ini menekankan pada karakteristik metodologis spesifik yang membedakannya dengan studi sebelumnya:

1. Penggunaan Local LLM: Berbeda dengan penelitian sebelumnya yang menggunakan API *cloud*, penelitian ini menggunakan arsitektur Phi-3 Mini [1] yang dijalankan secara lokal (*local LLM*). Pendekatan ini mengatasi masalah krusial terkait privasi dan kerahasiaan data historis korporat agensi digital yang tidak boleh terekspos ke *server* luar.
2. Validasi Silang pada Data Terbatas (LOOCV): Pada penelitian ini tidak menggunakan proporsi *train-test split* yang rentan bias pada dataset internal berskala kecil, melainkan menggunakan metode *Leave-One-Out Cross-Validation* (LOOCV) [8]. Hal ini memastikan setiap sampel data dievaluasi secara adil.
3. Strategi seleksi Few-Shot Berbasis Distribusi Batas: Penelitian terdahulu umumnya mengambil contoh *few-shot* secara acak. Penelitian ini memperkenalkan inovasi dengan memilih secara dinamis tiga contoh spesifik berdasarkan distribusi batas (*Min, Median, Max*) sebagai jangkar kognitif.

III. METODOLOGI

Desain penelitian ini dirancang secara eksperimental untuk mengevaluasi kinerja estimasi mandiri (*standalone*) dari *local* LLM. Alur kerja operasional penelitian secara menyeluruh dibagi ke dalam empat tahapan utama yang berfokus pada pemrosesan bahasa alami dan penalaran model terhadap data proyek perangkat lunak.



Gambar 1: Alur kerja operasional penelitian

A. Tahapan I - Pengumpulan dan Karakteristik Data

Penelitian ini menggunakan dataset historis sekunder berupa 20 portofolio proyek perangkat lunak dari sebuah digital agency di Bali. Meskipun ukuran populasi data ini tergolong kecil (*N*=20), sampel tersebut dinilai sangat representatif untuk lingkup *within-company* data karena secara utuh mencakup variabilitas karakteristik operasional, standar evaluasi pakar, serta tumpukan teknologi spesifik yang diaplikasikan di lingkungan agensi tersebut. Literatur SEE menunjukkan bahwa model estimasi yang dilatih menggunakan data internal perusahaan umumnya memiliki tingkat kesahihan ekologis yang jauh lebih tinggi dibandingkan pemanfaatan data silang-perusahaan (*cross-company data*) berskala masif yang sering kali rentan terhadap bias konteks [15],[17]. Lebih lanjut, keterbatasan jumlah sampel ini secara matematis telah dikompensasi melalui implementasi *Leave-One-Out Cross-Validation* (LOOCV), yang memastikan bahwa ke-20 proyek tersebut dievaluasi secara adil dan objektif tanpa mengurangi ukuran pool referensi data latih. Tingkat representasi sampel ini juga diperkuat oleh karakteristik variabel target yang digunakan, di mana nilai estimasi murni merepresentasikan keputusan kolektif *expert-agreed* baseline effort pada fase awal perencanaan, bukan waktu realisasi penyelesaian aktual di lapangan. Oleh karena itu, 20 dokumen spesifikasi kebutuhan ini sudah sangat memadai untuk memetakan pola evaluasi pakar tanpa terdistorsi oleh dinamika hambatan teknis saat konstruksi.

Fitur masukan atau variabel independen (X) yang diekstrak dari dataset ini tidak hanya bersandar pada aspek tekstual, melainkan dikombinasikan secara terstruktur menjadi tiga komponen utama, yaitu: teks deskripsi kebutuhan perangkat lunak (*Project_Description*) yang ditulis dalam bahasa alami, tipe platform pengembangan (*Platform_Type*), dan kategori tumpukan teknologi yang digunakan (*Tech_Stack_Category*). Sementara itu, variabel target atau variabel dependen (Y) yang digunakan adalah nilai estimasi upaya dasar historis (*Hist_Total_Effort*) yang diukur secara konsisten dalam satuan hari-kerja (*man-days*). Struktur representasi dari variabel input tekstual (X) beserta variabel target numerik (Y) dari dataset ini disajikan secara ringkas pada Tabel 1.

Perlu ditegaskan kembali bahwa variabel Y dalam penelitian ini merepresentasikan angka estimasi awal yang telah disetujui dan disepakati secara kolektif oleh tim ahli (*expert-agreed baseline effort*) sebelum tahapan konstruksi proyek dimulai. Variabel ini secara metodologis sengaja dipilih untuk menggantikan durasi atau realisasi waktu aktual penyelesaian proyek di lapangan. Pendekatan ini diterapkan untuk menjaga kesahihan konstruk (*construct validity*) dari pengujian model estimasi, sehingga hasil evaluasi murni mencerminkan kemampuan pemetaan model terhadap keputusan ideal para pakar tanpa terdistorsi oleh faktor eksternal di lapangan seperti hambatan teknis yang tidak terduga, perubahan ruang lingkup di tengah jalan (*scope creep*), maupun variasi tingkat produktivitas dari setiap pengembang individual.

TABEL I
STRUKTUR REPRESENTASI VARIABEL DAN CONTOH DATA

ID	Deskripsi Kebutuhan (Project_Description) Fitur Input (X)	Platform	Tech Stack	Estimasi Ahli Target (Y)
1	Proyek ini mencakup implementasi dan pembangunan total modul Company Profile yang terdiri dari: Home Page, About Us, Product, Sustainability, Project Gallery, dan Contact Us. Fokus utama adalah melakukan integrasi penuh terhadap desain yang sudah diserahkan untuk memastikan semua fungsionalitas...	Web	Wordpress	30 Hari
...	...	...	...	...
20	Proyek ini mencakup dua modul utama: User (untuk tampilan publik) dan Admin (untuk pengelolaan data event). Modul User akan menampilkan Halaman Utama, Widget Kalender, dan halaman kalender event lengkap dengan fitur pencarian, filter, dan desain *mobile responsive*. Modul Admin akan menyediakan fungsionalitas Login, pengelolaan data event (lihat, tambah, detail, update, hapus)...	Web	Laravel	90 Hari

Meskipun struktur dataset historis yang digunakan memiliki karakteristik yang ringkas serta murni mengombinasikan narasi kebutuhan tekstual dengan spesifikasi platform dan teknologi, sehingga karakteristik data ini memiliki kesahihan ekologis (*ecological validity*) yang sangat tinggi. Kondisi ini merepresentasikan realitas industri perangkat lunak yang sesungguhnya pada fase awal perencanaan proyek (*early-stage estimation*). Pada fase penting tersebut, seorang manajer

p-ISSN:1693 – 2951; e-ISSN: 2503-2372

proyek umumnya dituntut untuk merumuskan batas estimasi beban kerja dengan informasi spesifikasi yang masih sangat minim dan tidak terstruktur dari pihak klien. Bagi sebuah *Large Language Model* (LLM), kesederhanaan teks bahasa alami ini tetap menyimpan kekayaan konteks semantik yang mendalam. Arsitektur model mampu menangkap esensi tingkat kesulitan fitur secara langsung melalui mekanisme atensi internalnya tanpa memerlukan pemisahan atribut numerik yang rumit. Oleh karena itu, dataset ini valid dan sangat representatif untuk menguji kapabilitas penalaran mandiri model dalam kondisi riil di lapangan.

B. Tahapan II - Skenario Validasi Silang (LOOCV)

Mengingat karakteristik dataset historis tingkat korporat pada industri agensi digital lokal yang umumnya memiliki populasi sampel terbatas, pengujian model menggunakan metode pembagian data statis (seperti proporsi *training* dan *testing* sebesar 80:20) sangat tidak disarankan. Pendekatan konvensional tersebut berisiko tinggi memicu bias partisi, meningkatkan variansi galat, serta membatasi jumlah data referensi berharga yang dapat dipelajari oleh model. Oleh karena itu, penelitian ini mengimplementasikan metode validasi silang *Leave-One-Out Cross-Validation* (LOOCV) sebagai strategi evaluasi kinerja *local* LLM. Metode LOOCV merupakan bentuk khusus dari K*-fold cross-validation* di mana nilai ditetapkan sama dengan jumlah total sampel data (N). Pendekatan ini secara komputasi sangat efektif dan objektif untuk mengevaluasi data berskala kecil karena memaksimalkan pemanfaatan data historis yang tersedia sebagai basis pengetahuan emulasional model tanpa mengurangi porsi data uji [8].

Secara matematis, prosedur penanganan data dan mekanisme perputaran iterasi LOOCV dalam penelitian ini diformalisasikan secara sistematis. Misalkan terdapat sebuah himpunan dataset utama D yang memiliki total populasi sebanyak$N$proyek, yang dinyatakan dalam persamaan berikut:

$$D = \{p_1, p_2, p_3, \ldots, p_N\} \quad (1)$$

Setiap elemen sampel proyek p_i di dalam dataset merupakan pasangan terstruktur yang terdiri dari fitur input tekstual-teknis (X_i) dan nilai target numerik berupa estimasi upaya awal ahli (Y_i). Proses komputasi validasi silang ini dieksekusi secara sirkular sebanyak N kali iterasi. Pada setiap putaran iterasi ke-i (di mana $i = 1,2,\ldots,N$), dataset D dipartisi secara dinamis menjadi dua himpunan bagian yang sali*ng lepas (mutually* exclusive), yaitu:

1. Himpunan Data Uji Tunggal (D_{test}), yang mengisolasi tepat satu sampel proyek pada indeks ke-i sebagai objek evaluasi, sesuai dengan persamaan berikut:
$$D_{test} = \{p_1\} \quad (2)$$
2. Himpunan Data Historis Latih (D_{train}), yang menampung seluruh sisa sampel proyek sebanyak $N-1$sebagai *poo*l referensi kontekstual, diformalisasikan melalui persamaan berikut:
$$D_{train} = D \setminus \{p_i\} = \{j \neq i\} \quad (3)$$

Dari dalam *pool* data D_{train} yang berisi $N-1$ proyek tersebut, algoritma seleksi berbasis distribusi akan bekerja secara otomatis untuk menyaring tiga sampel proyek spesifik, yaitu proyek dengan nilai Y minimum (p_{min}), median (p_{median}), dan maksimum (p_{max}). Ketiga sampel berkarakteristik khusus ini ditransformasikan ke dalam representasi tekstual sebagai contoh *few-sho*t untuk memandu proses inferensi model terhadap data uji D_{test}.

Prosedur ini diulang secara konsisten hingga seluruh sampel proyek di dalam dataset D pernah diposisikan sebagai data uji tunggal tepat sebanyak satu kali. Hasil tebakan numerik yang diproduksi dari setiap putaran disimpan secara bertahap, sehingga pada akhir iterasi ke-N akan terbentuk sebuah himpunan nilai prediksi utuh $Y' = \{y_1\hat{}, y_2\hat{}, y_3\hat{}, \ldots, y_N\hat{}\}$. Himpunan nilai prediksi inilah yang kemudian diuji tingkat akurasi agregatnya terhadap himpunan nilai aktual kesepakatan ahli $Y = y_1, y_2, y_3, \ldots, y_N$ menggunakan kalkulasi metrik MMRE dan koefisien determinasi R^2[4].

C. Tahapan III - Strategi Prompt Engineering Terstruktur

Untuk menghasilkan prediksi estimasi upaya dari *local* LLM (menggunakan arsitektur Phi-3 Mini), penelitian ini tidak melakukan modifikasi bobot internal model (*fine-tuning*), melainkan murni memanfaatkan optimasi *Prompt Engineering*. Instruksi dirancang menggunakan kombinasi teknik *Few-Shot Prompting* dan *Chain-of-Thought* (CoT) [9],[10]. Mengambil contoh historis secara acak dari kumpulan data latih, penelitian ini memperkenalkan strategi seleksi contoh berbasis representasi distribusi batas (*distribution boundary representation*).

Pada setiap iterasi LOOCV, sistem secara dinamis memilih tepat tiga contoh proyek () dari kumpulan data historis yang memiliki karakteristik metrik beban kerja: terendah (min), nilai tengah (median), dan tertinggi (max). Ketiga contoh referensi ini tidak hanya memuat teks deskripsi proyek, tetapi juga diperkaya dengan informasi *Platfor*m dan *Tech Stac*k untuk memberikan konteks teknis yang holistik. Strategi penyediaan nilai batas bawah dan atas ini berfungsi sebagai jangkar kognitif *(cognitive anchor*) bagi LLM untuk memahami skala variansi proyek agensi, sekaligus meminimalisasi risiko halusinasi estimasi angka yang ekstrem.

Instruksi CoT diimplementasikan dengan memaksa model untuk tidak langsung menebak angka, melainkan melakukan komparasi relatif terlebih dahulu. Model diinstruksikan untuk memproduksi dua luaran secara berurutan. Pertama, teks "Penalaran" (*Reasoning*) dengan batasan ketat maksimal 30 kata. Pada tahap ini, model diwajibkan untuk membandingkan tingkat kompleksitas proyek yang sedang diuji (apakah lebih sederhana, setara, atau lebih kompleks) terhadap ketiga contoh historis yang diberikan. Kedua, "Estimasi Upaya" (*Estimated Effort*) berupa nilai numerik tunggal yang merupakan hasil deduksi dari penalaran

komparatif tersebut. Batasan jumlah kata diterapkan secara empiris untuk menjaga efisiensi komputasi *token* lokal sekaligus mereduksi kecenderungan model berhalusinasi di luar konteks.

Untuk menjamin reproduktibilitas penelitian, dekonstruksi dari templat *prompt* terstruktur yang disuplai ke dalam LLM disajikan pada Tabel 2.

TABEL II
DEKONSTRUKSI STRUKTUR PROMPT ENGINEERING (COT & FEW-SHOT)

Komponen Prompt	**Isi Instruksi (Bahasa Inggris)**	**Fungsi Kognitif**
Instruksi Sistem (System Prompt)	*"You are an expert Software Project Estimator. Analyze the requirements, platform, and tech stack to predict the development effort in Man-Days. Calculate your estimate by comparing the current project's complexity to the provided historical examples. First, write your reasoning (max 30 words). Explicitly state if this project is simpler, similar, or more complex than the specific examples. Then, provide the final numerical estimate based on that relative comparison. STRICT OUTPUT FORMAT: Reasoning: [...] Estimated Effort: [...]"*	Mendefinisikan persona model, memberikan aturan komparasi relatif, membatasi panjang teks alasan, dan mengunci format luaran.
Konteks Historis (*Few-Shot Examples*)	*"### HISTORICAL EXAMPLES ### # Example 1 (Min) Platform: [Platform Type] Tech Stack: [Tech Stack] Description: [Deskripsi Proyek] Baseline Effort: [Angka] Man-Days (Diikuti oleh Example 2 (Median) dan Example 3 (Max))"*	Menyediakan jangkar distribusi (batas bawah, tengah, atas) beserta konteks spesifikasi teknologi sebagai referensi perbandingan.
Data Uji (*Current Project*)	*"### CURRENT PROJECT TO ESTIMATE ### Platform: [Platform Data Uji] Tech Stack: [Tech Stack Data Uji] Description: [Deskripsi Data Uji] Based on the historical examples, compare the current project's complexity to the examples, then provide your reasoning and estimate."*	Menginjeksikan data proyek yang akan diestimasi dan memicu eksekusi penalaran komparatif.

D. Tahapan IV - Evaluasi Kinerja

Kumpulan nilai prediksi numerik yang berhasil diekstrak dari keseluruhan iterasi LOOCV diagregasi untuk dievaluasi kinerjanya secara kuantitatif. Sesuai dengan standar evaluasi metrik pada *Software Effort Estimation* (SEE), akurasi dan keandalan model diukur menggunakan dua parameter utama, yaitu *Mean Magnitude of Relative Error* (MMRE), Koefisien Determinasi (R*-squared*), dan PRED(25) [4].

Metrik MMRE digunakan untuk menghitung rata-rata akumulatif dari persentase galat (kesalahan) relatif antara nilai prediksi model terhadap nilai kesepakatan ahli. Evaluasi menggunakan MMRE memberikan gambaran objektif mengenai tingkat presisi prediksi secara keseluruhan, di mana nilai yang semakin kecil atau mendekati nol mengindikasikan kesalahan estimasi yang semakin rendah. Meskipun beberapa studi terdahulu menyoroti bahwa MMRE dapat memberikan bias terhadap prediksi yang terlalu rendah (*underestimates*) [11], metrik ini tetap dipertahankan dalam pengujian ini karena MMRE merupakan parameter evaluasi paling universal dan diakui secara luas dalam literatur SEE untuk menakar magnitudo galat[20].

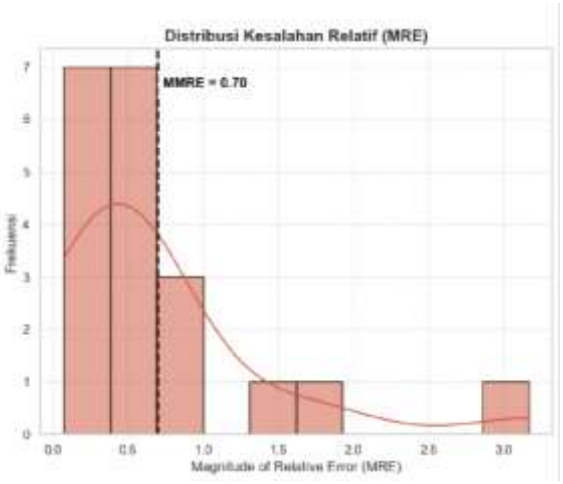
Sementara itu, metrik R^2 bertindak sebagai indikator *goodness-of-fit* untuk mengukur seberapa baik variabilitas nilai estimasi awal dari tim ahli dapat dijelaskan oleh model prediktif LLM melalui pendekatan tekstualnya. Nilai R^2 memiliki rentang antara $-\infty$ hingga 1. Nilai yang mendekati 1 menandakan pola prediksi model selaras secara kuat dengan sebaran data historis agensi. Sebaliknya, nilai R^2 yang bernilai negatif menunjukkan sebuah temuan empiris bahwa kinerja penalaran numerik model memiliki penyimpangan yang besar dan menghasilkan prediksi yang lebih buruk dibandingkan dengan hanya menggunakan nilai rata-rata (*mean*) dari dataset historis tersebut.

Selain MMRE dan R^2, penelitian ini juga menggunakan metrik PRED(25) untuk memberikan gambaran akurasi yang lebih komprehensif dalam skala industri. PRED(25) mengukur persentase prediksi yang memiliki nilai *Relative Error* (RE) kurang dari atau sama dengan 0,25. Metrik ini penting untuk menunjukkan proporsi estimasi yang dianggap sukses atau dapat diterima menurut standar industri rekayasa perangkat lunak, meskipun nilai galat rata-rata (MMRE) secara keseluruhan mungkin tinggi akibat adanya pencilan.

IV. HASIL DAN PEMBAHASAN

A. Evaluasi Kinerja Kuantitatif

Kinerja estimasi mandiri dari model bahasa Phi-3 Mini dievaluasi secara kuantitatif menggunakan metrik *Mean Magnitude of Relative Error* (MMRE), *PRED*(25) dan Koefisien Determinasi (R^2). Hasil agregasi dari keseluruhan iterasi pengujian *Leave-One-Out Cross-Validation* (LOOCV) divisualisasikan pada Gambar 2. Secara statistik, model menghasilkan nilai tingkat kesalahan prediksi (MMRE) sebesar 0,6963 atau setara dengan rasio galat 69,63%. Tingginya nilai MMRE ini didukung oleh sebaran distribusi kesalahan relatif yang sangat lebar pada histogram galat.



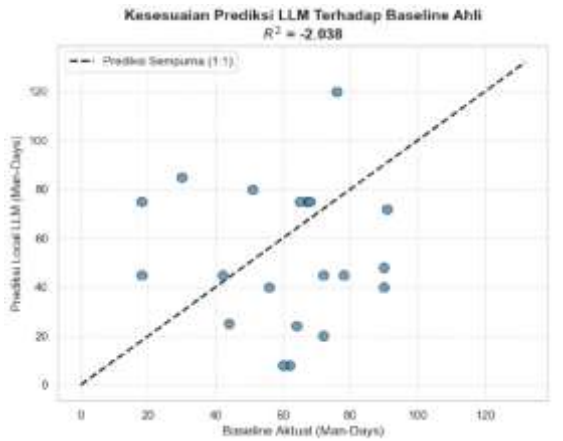
Gambar 2: Distribusi Kesalahan Relatif

Lebih lanjut, evaluasi kecocokan variansi model menghasilkan nilai R^2 sebesar -2,038. Perolehan nilai R^2 yang negatif secara matematis mengindikasikan bahwa luaran prediksi numerik dari *local* LLM memiliki tingkat deviasi yang lebih buruk jika dibandingkan dengan sekadar menggunakan tebakan nilai rata-rata (*mean*) dari kumpulan data historis itu sendiri.

TABEL III
PERBANDINGAN PERFORMA PREDIKSI

Metode	MMRE	PRED(25)	R-Squared
Mean Baseline	0.4470	-	0,000
LLM	0.6963	0.25	-2,038

Berdasarkan Tabel III, terlihat bahwa performa numerik Phi-3 Mini secara mandiri (MMRE 0,69) masih berada di bawah performa *Mean Baseline* (MMRE 0,44). Hal ini menjelaskan mengapa nilai R^2 yang dihasilkan bernilai negatif (-2,038). Meskipun demikian, metrik PRED(25) menunjukkan bahwa 25% dari total prediksi model telah mencapai tingkat akurasi industri (galat $\leq$ 25%). Ketidakmampuan model untuk melampaui baseline rata-rata disebabkan oleh beberapa kasus halusinasi numerik yang ekstrem, seperti pada proyek 'e-commerce Laravel' di mana model memberikan prediksi yang sangat rendah (8 *man-days*) meskipun penalaran logisnya sudah tepat secara semantik.



Gambar 3: Kesesuaian Prediksi LLM Terhadap Baseline Ahli

Sebaran *scatter plot* juga memperlihatkan bahwa model sering kali terjebak pada angka tebakan spesifik (seperti 8, 40, atau 75) yang menyimpang jauh dari garis prediksi ideal.

Selain evaluasi numerik, penelitian ini memperluas cakupan analisis dengan meninjau kemampuan model dalam melakukan klasifikasi tingkat kompleksitas (*Low, Medium, High*). Berdasarkan data hasil pengujian, ditemukan bahwa meskipun model sering mengalami halusinasi pada angka eksak, model memiliki tingkat akurasi yang lebih stabil dalam mengidentifikasi kategori beban kerja proyek. Sebagai contoh, pada proyek dengan kategori aktual '*Medium*', model berhasil memberikan label klasifikasi yang tepat meskipun nilai hari-kerja yang diprediksi mengalami penyimpangan. Hal ini mengindikasikan bahwa LLM lebih handal sebagai penilai kualitatif dibandingkan sebagai kalkulator regresi numerik.

B. Analisis Kualitatif Penalaran Model

Meskipun evaluasi numerik menunjukkan kinerja yang suboptimal, analisis mendalam terhadap luaran teks penalaran (*Reasoning*) yang diekstrak dari instruksi *Chain-of-Thought* mengungkap fenomena *bottleneck* kognitif yang unik pada arsitektur model bahasa. Model secara konsisten mampu mengeksekusi analisis semantik dan komparasi relatif dengan sangat baik, namun gagal menerjemahkan logika tersebut ke dalam ruang dimensi kontinu numerik (*numerical regression*).

Sebagai studi kasus yang diekstrak dari luaran prediksi, pada pengujian proyek "pembaruan (*revamp*) website akademi" (nilai aktual 18 *man-days*), model berhasil mengidentifikasi bahwa proyek tersebut berbasis *Wordpress* dan secara logika *"less complex than Examples with Laravel and additional modules"*.

Secara semantik, penalaran ini sangat akurat dan sesuai dengan pakem industri rekayasa perangkat lunak. Namun, model mengalami halusinasi regresi dengan menyimpulkan angka akhir sebesar 75 *man-days*, yang justru lebih tinggi dari sebagian besar proyek kompleks di dalam memori historisnya. Pada kasus lain, seperti proyek modul e-commerce berbasis *Laravel* (nilai aktual 62 *man-days*), model dengan tajam menalar bahwa proyek tersebut memiliki *"fewer features... likely less effort required"* dibandingkan contoh ekstrem tertinggi. Sekali lagi, logika penalarannya valid, tetapi model melakukan penalti numerik yang terlalu ekstrem dengan menebak angka 8 *man-days*.

Fenomena ini selaras dengan tinjauan literatur terkini yang menemukan bahwa, meskipun LLM memiliki keunggulan luar biasa dalam rekayasa kebutuhan (*requirements engineering*) dan analisis kode, arsitektur utamanya sering kali mengalami keterbatasan kognitif (*hallucinations*) ketika dipaksa menyelesaikan tugas penalaran aritmetika dan regresi skala kontinu secara pasti [12],[13].

C. Implikasi terhadap Estimasi Perangkat Lunak

Temuan empiris ini memberikan wawasan krusial bagi literatur *Software Effort Estimation* (SEE). Penggunaan instruksi *Chain-of-Thought* terbukti berhasil mereduksi halusinasi tekstual dan memandu model untuk membandingkan spesifikasi platform dan tumpukan teknologi secara presisi. Namun, membebani *local* LLM untuk langsung menebak angka estimasi secara mandiri (*standalone*) menghasilkan bias interpolasi yang signifikan. Model bahasa dilatih untuk

memprediksi token kata berdasarkan distribusi probabilitas teks, bukan mengkalkulasi regresi matematis dari variabel target.

Penelitian ini mengonfirmasi bahwa kapabilitas terbaik LLM berskala kecil di lingkungan agensi digital terletak pada kemampuannya untuk mengekstraksi tingkat kompleksitas kualitatif dari dokumen kebutuhan bahasa alami yang tidak terstruktur. Untuk mencapai tingkat akurasi numerik yang relevan dengan industri, luaran penalaran dari model ini tidak dapat berdiri sendiri, melainkan memerlukan integrasi atau hibridisasi dengan algoritma pemodelan statistik konvensional.

V. KESIMPULAN

Penelitian ini merupakan studi eksplorasi awal dan mengungkap potensi besar dari pemanfaatan kecerdasan buatan generatif berskala lokal (*local* LLM) dengan arsitektur Phi-3 Mini pada fase awal perencanaan proyek perangkat lunak. Melalui penerapan optimasi *Prompt Engineering* berbasis *Chain-of-Thought* (CoT), analisis kualitatif terhadap luaran penalaran (*reasoning*) membuktikan bahwa model bahasa memiliki kecerdasan semantik yang sangat tinggi dalam membedah dokumen kebutuhan proyek yang tidak terstruktur. Model secara konsisten mampu mengidentifikasi pengaruh tipe platform dan tumpukan teknologi (*tech stack*) terhadap skala proyek, serta mampu membandingkan tingkat kompleksitas relatif antara satu proyek dengan proyek historis lainnya secara logis dan terstruktur.

Untuk memaksimalkan keunggulan tersebut dan mencapai tingkat akurasi *Software Effort Estimation* (SEE) yang dapat diandalkan, penelitian ini secara kuat merekomendasikan pengembangan kerangka kerja estimasi hibrida. Dalam kerangka kerja ini, kemampuan analitis LLM diposisikan murni sebagai pengekstraksi fitur kualitatif (*qualitative feature extractor*). Hasil penalaran semantik tersebut kemudian diumpankan sebagai variabel masukan ke dalam algoritma *Machine Learning* konvensional (seperti *Random Forest* atau regresi ansambel lainnya) yang dirancang khusus untuk menangani kalkulasi regresi numerik secara presisi[17].

Rekomendasi pendekatan hibrida ini didasarkan pada temuan empiris pengujian *Leave-One-Out Cross-Validation* (LOOCV), yang menyimpulkan bahwa *local* LLM tidak direkomendasikan untuk digunakan sebagai penaksir estimasi numerik secara mandiri (*standalone*). Evaluasi kuantitatif menunjukkan bahwa model bahasa memiliki keterbatasan fundamental dalam melakukan tugas regresi matematis, dibuktikan oleh perolehan nilai *Mean Magnitude of Relative Error* (MMRE) yang tinggi serta Koefisien Determinasi (R^2) yang bernilai negatif. Keterbatasan ini menegaskan bahwa LLM cenderung mengalami bias interpolasi dan halusinasi numerik saat menebak angka eksak, sehingga penggabungannya dengan *Machine Learning* menjadi solusi yang paling optimal.

REFERENSI

[1] M. Abdin, et al., "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[2] C. N. Coelho Jr, et al., "Effort and size estimation in software projects with large language model-based intelligent interfaces," *arXiv preprint arXiv:2402.07158*, 2024.

[3] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 37-60, 2004.

[4] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation," *arXiv preprint arXiv:2101.10658*, 2021.

[5] Ł. Radliński and J. Swacha, "Large language models for early-stage software project estimation: A systematic mapping study," *Applied Sciences*, vol. 15, no. 24, p. 13099, 2025.

[6] S. A. Salihu, K. B. Saliu, and O. A. Owoyemi, "A systematic literature review of machine learning and AutoML in software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 8, 2024.

[7] N. Tran, T. Tran, and N. Nguyen, "Leveraging AI for enhanced software effort estimation: A comprehensive study and framework proposal," *arXiv preprint arXiv:2402.05484*, 2024.

[8] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 1403-1416, 2012.

[9] J. Wei, et al., "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824-24837, 2022.

[10] T. Brown, et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877-1901, 2020.

[11] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Transactions on Software Engineering*, vol. 29, no. 11, pp. 985-995, Nov. 2003.

[12] X. Hou, et al., "Large language models for software engineering: A systematic literature review," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 5, pp. 1-41, 2023.

[13] A. Fan, et al., "Large language models for software engineering: Survey and open problems," *arXiv preprint arXiv:2310.03533*, 2023.

[14] B. W. Boehm, *Software engineering economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.

[15] B. A. Kitchenham, E. Mendes, and G. H. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Transactions on Software Engineering*, vol. 28, no. 5, pp. 316-329, 2002.

[16] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, vol. 54, no. 1, pp. 41-59, 2012.

[17] I. P. S. Handika, I. A. Giriantari, and A. Dharma, "Perbandingan Metode Extreme Learning Machine dan Particle Swarm Optimization Extreme Learning Machine untuk Peramalan Jumlah Penjualan Barang," *Majalah Ilmiah Teknologi Elektro*, vol. 15, no. 1, pp. 84-90, 2016.

[18] N. L. K. T. W. M.U, I. M. O. Widyantara, and N. M. A. E. Dewi W, "Deteksi Tipe Modulasi Digital Pada Automatic Modulation Recognition Menggunakan Support Vector Machine dan Conjugate Gradient Polak Ribiere-Backpropagation," *Majalah Ilmiah Teknologi Elektro*, vol. 18, no. 2, pp. 281-286, 2019.

[19] T. Kojima, A. Gu, M. Reid, Y. Matsuo, and S. Iwasawa, "Large language models are zero-shot reasoners," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22199-22213, 2022.

[20] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 380-391, 2005.

{Halaman ini sengaja dikosongkan}