ARSITEKTUR WORKSPACE-FIRMWARE DAN MANDAT EKSEKUSI PADA AGENTIC AI LOKAL

Yafie Miftah Imani[1], Tiyan Ganang Wicaksono[2]

[12,]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Duta Bangsa

Jalan Bhayangkara No.55, Tipes, Kec. Serengan, Kota Surakarta, Jawa Tengah 57154
[1]yafie_miftahimani@udb.ac.id, [2]tiyan_ganang@udb.ac.id

Abstract

The implementation of Small Language Models (sLLM) as standalone agents on local infrastructure offers efficiency and data privacy, but is often hampered by communication constraints that lead to execution paralysis. This study proposes a Workspace-Firmware architecture and an "Execution Mandate" strategy to optimize the capabilities of intelligent agents (Agentic AI) using the Qwen 1.5B model within the OpenClaw framework. The research method is carried out through a three-stage experiment, including testing functional instruction compliance, session memory continuity, and problem-handling automation on a Windows/XAMPP local server environment. The evaluation results show that the Execution Mandate strategy successfully triggers external command invocation functions (tool-calling). However, the small-scale model (1.5B) was found to still experience systemic obstacles in the form of environmental errors (Environment Amnesia, such as confusion of Linux directory paths on Windows systems) and command syntax errors (Syntax Hallucination in the form of mixing instructions between bash and Windows CMD). Furthermore, the performance of these intelligent agents is heavily influenced by the speed of the physical hardware, with HDD storage being the primary bottleneck in the agent's memory data read process. This study concludes that small-scale sLLMs are suitable for specific standalone tasks, but require additional middleware guardrails and SSD specification standardization to handle complex server automation (DevOps) workflows.

Keywords : *Agentic AI, Small Language Models, Workspace-Firmware, Execution Mandate, DevOps Automation*

Abstrak

Implementasi Model Bahasa Kecil atau *Small Language Models (sLLM)* sebagai agen mandiri pada infrastruktur lokal menawarkan efisiensi dan privasi data, namun sering kali terhambat oleh kendala komunikasi yang menyebabkan kegagalan eksekusi perintah (*execution paralysis*). Penelitian ini mengusulkan arsitektur *Workspace-Firmware* serta strategi "Mandat Eksekusi" untuk mengoptimalkan kemampuan agen pintar (Agentic AI) menggunakan model Qwen 1.5B dalam kerangka kerja OpenClaw. Metode penelitian dilakukan melalui eksperimen tiga tahap, meliputi pengujian kepatuhan instruksi fungsional, kesinambungan memori sesi, dan otomatisasi penanganan masalah pada lingkungan server lokal Windows/XAMPP. Hasil evaluasi menunjukkan bahwa strategi Mandat Eksekusi berhasil memicu fungsi pemanggilan perintah luar (*tool-calling*). Meskipun demikian, model berskala kecil (1.5B) ditemukan masih mengalami kendala sistemik berupa kesalahan pemahaman lingkungan (*Environment Amnesia*, seperti kerancuan jalur direktori Linux pada sistem Windows) dan kesalahan sintaksis perintah (*Syntax Hallucination* berupa percampuran instruksi antara bash dan Windows CMD). Selain itu, performa agen pintar ini sangat dipengaruhi oleh kecepatan perangkat keras fisik, di mana penggunaan media penyimpanan HDD menjadi hambatan (*bottleneck*) utama dalam proses pembacaan data memori agen. Penelitian ini menyimpulkan bahwa sLLM berukuran kecil layak digunakan untuk tugas-tugas mandiri yang spesifik, namun memerlukan sistem pengaman tambahan (*middleware guardrails*) serta standarisasi spesifikasi SSD untuk menangani alur kerja otomatisasi server (DevOps) yang kompleks.

Kata kunci : *Agentic AI, Small Language Models, Workspace-Firmware, Mandat Eksekusi, DevOps Automation.*

1. PENDAHULUAN

Perkembangan teknologi Kecerdasan Buatan (AI) telah memasuki fase transformatif dengan munculnya paradigma Agentic AI, di mana model bahasa besar (*Large Language Models - LLM*) tidak lagi sekadar berfungsi sebagai sistem pemroses teks statis, melainkan sebagai agen otonom yang mampu menyelesaikan tujuan yang kompleks dan berlapis, melakukan perencanaan, penalaran, dan eksekusi tugas dalam lingkungan komputasi nyata [1], [2], [3]. Pendekatan berbasis agen cerdas dapat membantu mengurangi kesalahan manusia sekaligus mempercepat penyelesaian tugas yang sebelumnya dilakukan secara manual [4]. Pemanfaatan Natural Language Processing (NLP) dalam mengolah teks menjadi salah satu pilih yang telah banyak diimplementasikan pada sistem berbasis aturan dan *Retrieval-Augmented Generation (RAG)* . Namun, pada penelitian ini, kemampuan NLP tidak hanya digunakan untuk menjawab pertanyaan, melainkan ditingkatkan untuk memicu pemanggilan alat (tool-calling) pada sistem operasi lokal melalui sLLM. Autonomous AI agents merupakan entitas cerdas yang memanfaatkan model bahasa untuk melakukan tugas kompleks secara mandiri, namun penerapannya tetap menghadapi tantangan terkait etika, penjelasan model, dan keamanan system [5]. Namun, ketergantungan pada model parameter besar berbasis cloud sering kali memunculkan kendala terkait latensi jaringan, biaya operasional yang tinggi, dan risiko kebocoran data sensitif organisasi. Sebagai solusi, penggunaan *Small Language Models* (sLLM) seperti Qwen2.5:1.5B menjadi alternatif yang menjanjikan karena efisiensinya dalam pemrosesan lokal dan kemampuan untuk menjaga kedaulatan data [6].

Meskipun sLLM menawarkan efisiensi sumber daya, implementasinya sebagai agen otonom menghadapi tantangan fundamental yang dikenal sebagai "kelumpuhan eksekusi" (*execution paralysis*). Masalah ini terjadi ketika model AI gagal membedakan antara perannya sebagai asisten percakapan dan perannya sebagai operator sistem. Dalam banyak kasus, model cenderung memberikan respon berupa tutorial atau instruksi teks manual alih-alih melakukan tindakan sistemik secara langsung. Fenomena ini berakar pada training objective model yang umumnya dioptimalkan melalui *Reinforcement Learning from Human Feedback* (RLHF) untuk menjadi asisten yang informatif, sehingga menciptakan bias perilaku yang menghindari pengambilan risiko tindakan sistemik [7]. Tanpa intervensi yang tepat, potensi sLLM untuk otomatisasi DevOps lokal akan terhambat oleh kecenderungan model untuk sekadar "berbicara" daripada "bertindak".

Dalam beberapa tahun terakhir, fokus riset kecerdasan buatan mulai bergeser ke arah efisiensi komputasi melalui pemanfaatan *Small Language Model (sLLM)*. Fenomena ini dibuktikan dengan munculnya model-model berukuran kecil yang mampu menunjukkan kapabilitas kognitif tinggi dan dapat dioperasikan secara lokal pada perangkat dengan spesifikasi terbatas, bahkan hingga level komputasi mobile [8]. Selain kendala pada perilaku model, arsitektur integrasi agen pada framework otonom saat ini masih didominasi oleh penggunaan konfigurasi JSON statis yang kaku. Pendekatan konvensional ini memiliki skalabilitas yang rendah dan menyulitkan pengembang dalam melakukan pembaruan protokol kerja agen secara dinamis. Terdapat celah penelitian (*research gap*) mengenai bagaimana menginjeksi kapabilitas operasional ke dalam sLLM secara fleksibel tanpa harus melalui proses *fine-tuning* yang kompleks. Penelitian sebelumnya dalam lingkup sistem informasi telah menekankan pentingnya efisiensi dalam otomatisasi data untuk mereduksi *human error* [9], namun penelitian yang secara spesifik membahas orkestrasi sistem lokal menggunakan sLLM dengan pendekatan firmware berbasis teks masih sangat terbatas.

Penelitian ini menempatkan arsitektur eksekusi dan manajemen konteks sebagai faktor penting dalam meningkatkan performa sistem AI otonom [10] dengan mengusulkan sebuah metodologi inovatif melalui pemanfaatan framework *OpenClaw* yang diintegrasikan dengan arsitektur "*Workspace-based Firmware*". Berbeda dengan metode konfigurasi konvensional, arsitektur ini menanamkan instruksi operasional, batasan etika, dan pemetaan keahlian agen melalui sekumpulan file Markdown terstruktur, yaitu *SOUL.md, AGENTS.md, dan SKILL.md.* Pendekatan ini mengadopsi prinsip fleksibilitas instruksi dalam memori kerja agen, yang memungkinkan agen untuk memiliki persona dan set keterampilan yang dapat diperbarui secara instan melalui manipulasi file di dalam lingkungan kerja (*workspace*). Hal ini selaras dengan konsep *Generative Agents* yang menggunakan arsitektur memori untuk mengatur perilaku kompleks dalam lingkungan simulasi [11].

Untuk mengatasi masalah *execution paralysis*, penelitian ini juga memperkenalkan strategi "Mandat Eksekusi" (*Execution Mandate*). Strategi ini merupakan teknik *hard-prompting* yang dirancang untuk merekayasa ulang prioritas keputusan pada sLLM dengan cara mengklasifikasikan pemberian *tutorial* sebagai bentuk kegagalan sistem (*system failure*). Dengan memberikan tekanan instruksional

bahwa hanya tindakan sistemik (seperti penulisan file, eksekusi terminal, atau kontrol layanan server) yang dianggap sebagai indikator kompetensi, model dipaksa untuk memicu mekanisme *tool-calling* secara proaktif [12]. Skenario uji coba difokuskan pada tugas-tugas kritis administrasi IT lokal, termasuk manajemen file *.json* dan aktivasi layanan server XAMPP, yang memerlukan ketepatan eksekusi tanpa intervensi manual yang repetitif.

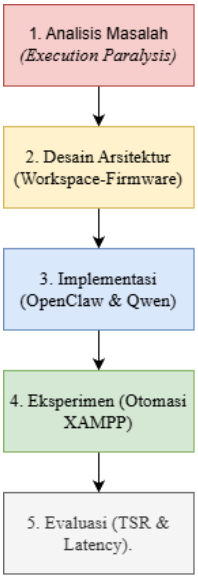
Tujuan utama dari penelitian ini adalah untuk mengevaluasi sejauh mana penggabungan arsitektur *Workspace-based Firmware* dan strategi Mandat Eksekusi dapat meningkatkan tingkat keberhasilan tugas (*task success rate*) pada model Qwen2.5:1.5B dalam lingkungan *OpenClaw*. Kontribusi ilmiah dari penelitian ini diharapkan dapat memberikan kerangka kerja baru bagi pengembangan Agentic AI lokal yang ringan, mandiri, dan handal untuk mendukung keberlanjutan operasional sistem informasi di berbagai skala organisasi.

2. METODOLOGI PENELITIAN

Metodologi yang diterapkan dalam penelitian ini menggunakan pendekatan eksperimental laboratoris yang berfokus pada pengembangan purwarupa (*prototyping*) sistem agen otonom. Penelitian ini dirancang untuk menguji fungsionalitas dan proaktifitas *Small Language Models* (sLLM) dalam mengelola instruksi administratif pada sistem operasi lokal melalui *framework OpenClaw*. Tahapan metodologi disusun secara sistematis guna memastikan replikasi penelitian dan validitas data yang dihasilkan.

2.1. Alur Penelitian

Penelitian ini mengikuti siklus pengembangan sistem yang terstruktur untuk menjamin konsistensi hasil. Alur penelitian secara visual direpresentasikan dalam Gambar 1, yang menggambarkan transisi dari tahap identifikasi masalah hingga evaluasi performa agen.



Gambar 1. Alur Penelitian

Tahap pertama adalah Identifikasi dan Analisis Masalah, di mana keterbatasan sLLM dalam mengeksekusi perintah sistem diidentifikasi sebagai *Execution Paralysis*. Tahap kedua adalah Desain Arsitektur Sistem, yang mencakup perancangan struktur *Workspace-based Firmware*. Tahap ketiga adalah Implementasi Teknikal, yaitu pengintegrasian model Qwen2.5:1.5B ke dalam *framework OpenClaw* dan penulisan protokol instruksi pada file Markdown. Tahap keempat adalah Eksperimen dan Pengujian, yang melibatkan skenario otomasi *DevOps* pada layanan XAMPP. Tahap terakhir adalah Evaluasi dan Penarikan Kesimpulan, di mana hasil eksekusi agen dianalisis menggunakan metrik kuantitatif.

Guna meningkatkan keterbacaan dan memberikan gambaran ringkas mengenai parameter operasional di setiap tahapan, ringkasan alur metodologi penelitian ini dirangkum secara detail pada Tabel I.

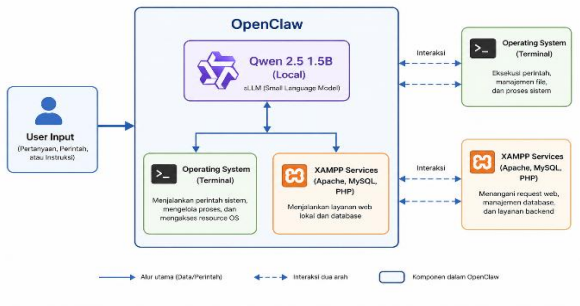
TABEL I. RINGKASAN ALUR METODOLOGI PENELITIAN

Tahap Penelitian	Aktivitas Utama	Instrumen / Komponen Terkait	Output/Hasil Tahap
1. Analisis Masalah	Mengidentifikasi akar penyebab kegagalan sLLM dalam mengeksekusi perintah sistem langsung.	Study literatur & observasi perilaku awal openclaw	Definisi Batasan *execution paralysis* pada model kecil.
2. Desain Arsitektur	Merancang konsep injeksi konteks berlapis untuk mengarahkan	Cetak biru arstitektur *Workspace-Fimware*	Spesifikasi logika pemisahan berkas *Sovereign DevOps.*

	probabilitas token model.		
3. Implementasi	Konfigurasi keterhubungan model Bahasa dengan kerangka kerja eksekusi terminal lokal.	Qwen 2.5 1.5B, Openclaw, Phyton 3.10+	File repositori firmware aktif (Soul.md, Agents.md, dan Skill.md)
4. Eksperimen	Pengujian fungsionalitas agen terhadap skenario penanganan masalah administratif server.	OS Windows, lingkungan local xampp (Apache/MySQL)	Berkas log eksekusi terminal dan data riwayat *tool-calling*.
5. Evaluasi	Pengukuran performa kognitif agen berdasarkan Tingkat keberhasilan dan kecepatan waktu respons.	Formulasi metrik kuantitatif TSR dan analisis latensi	Penilaian kelayakan sLLM, pemetaan anomali kognitif dan rekomendasi infrastruktur.

2.2. Spesifikasi Infrastruktur dan Arsitektur Sistem

Keberhasilan implementasi AI lokal tidak hanya ditentukan oleh kemampuan model, melainkan juga oleh kualitas orkestrasi eksekusi dan stabilitas infrastruktur pendukungnya [13], [14]. Kemampuan model *Deep Learning* dalam mengekstraksi dan mengklasifikasikan entitas spesifik dari input bahasa alami telah terbukti lebih unggul dibandingkan metode komputasi tradisional [15]. Dalam konteks Agentic AI, kemampuan ekstraksi entitas ini menjadi krusial agar model dapat menerjemahkan frasa percakapan pengguna menjadi parameter eksekusi terminal yang presisi. Akurasi dan kecepatan respon agen sangat bergantung pada infrastruktur komputasi yang digunakan. Dalam penelitian ini, system dijalankan pada lingkungan lokal yang terintegrasi. Arsitektur teknis sistem ditunjukkan pada Gambar 2, yang memperlihatkan bagaimana framework OpenClaw bertindak sebagai jembatan kognitif.



Gambar 2. Arsitektur Integrasi OpenClaw dan sLLM Lokal

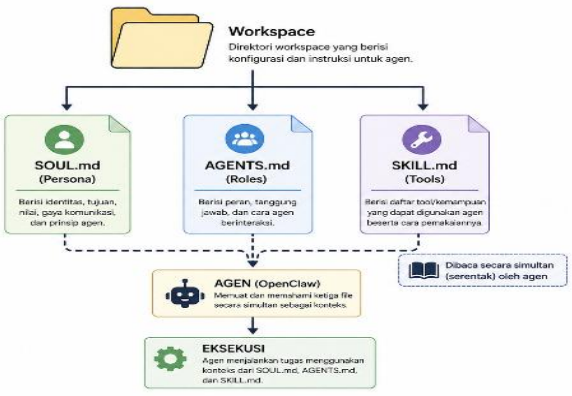
Spesifikasi komponen utama adalah sebagai berikut:

1. Model Bahasa (*Core AI*): Qwen 2.5 dengan parameter 1.5B. Pemilihan model ini didasarkan pada efisiensi rasio parameter ke performa yang optimal untuk *edge computing* dan pemrosesan lokal [6].
2. *Middleware Framework*: *OpenClaw*, yang berfungsi sebagai jembatan antara instruksi bahasa alami dan antarmuka pemrograman aplikasi (API) sistem operasi.
3. *Environment*: Python 3.10+ dengan pustaka pendukung untuk eksekusi terminal dan manajemen proses.
4. Target Sistem: Server lokal XAMPP (Apache dan MySQL) untuk pengujian kontrol layanan.

2.3. Perancangan *Workspace-based Firmware*

Proses transformasi dari instruksi bahasa alami menjadi perintah terminal pada arsitektur yang diusulkan melibatkan alur kerja *multi-layer context injection*. Ketika pengguna memasukkan perintah, sistem tidak langsung meneruskannya ke model bahasa. Sebaliknya, *framework OpenClaw* melakukan pemindaian terhadap berkas *firmware* statis (*SOUL.md*) untuk menyuntikkan batasan etika dan perilaku, kemudian memadukannya dengan status memori terkini dari *AGENTS.md*. Logika ini dirancang sebagai lapisan middleware kognitif yang berfungsi menyaring ambiguitas.

Setiap token yang dihasilkan oleh sLLM Qwen 1.5B kemudian dipetakan ke dalam skema *tool-calling* yang telah didefinisikan dalam *TOOLS.md.* Jika model mendeteksi kebutuhan akan akses sistem, ia akan memicu fungsi *execute_command* atau *write_file*. Keunggulan dari logika ini adalah kemampuannya untuk mengarahkan probabilitas token model menjauh dari mode percakapan umum menuju mode tindakan teknis. Alur ini memastikan bahwa setiap tindakan yang diambil oleh agen memiliki landasan kontekstual yang kuat dari riwayat sesi sebelumnya, sehingga memitigasi risiko tindakan yang redundan atau tidak relevan dengan kondisi peladen saat itu.



Gambar 3. Hierarki File Workspace-based Firmware

2.3.1. *SOUL.md*: Inti Persona dan Batasan Etika *File*

SOUL.md berfungsi sebagai jangkar kepribadian agen. Di dalamnya didefinisikan bahwa agen bukan hanya asisten percakapan, melainkan seorang "*Sovereign DevOps Engineer*". Instruksi di dalam file ini mencakup batasan etika, gaya bahasa yang teknis, dan objektif jangka panjang agen untuk menekan bias respons tutorial [11].

2.3.2. *AGENTS.md*: Manajemen Peran dan Hierarki Tugas

AGENTS.md digunakan untuk membagi kapabilitas model ke dalam sub-peran. Sebagai contoh, agen dapat beralih peran menjadi "*System Administrator*" saat mengelola server. Hierarki ini memungkinkan model sLLM yang memiliki keterbatasan jendela konteks *(context window*) untuk tetap fokus pada instruksi spesifik yang relevan.

2.3.3. *SKILL.md*: Pemetaan Kemampuan dan *Tool-Calling*

File *SKILL.md* bertindak sebagai dokumentasi teknis yang mencakup daftar perintah terminal yang diizinkan (seperti *mkdir*, *touch*, atau perintah kontrol XAMPP). Pendekatan ini selaras dengan konsep *Toolformer*, di mana model belajar menggunakan alat eksternal untuk menyelesaikan tugas yang tidak dapat diselesaikan melalui teks semata [12].

2.4. Implementasi Strategi Mandat Eksekusi (*Execution Mandate*)

Guna mengatasi masalah *Execution Paralysis*, penelitian ini menerapkan strategi "Mandat Eksekusi". Secara teknis, strategi ini dilakukan dengan menyisipkan instruksi tingkat tinggi ke dalam *system prompt* yang secara eksplisit menyatakan bahwa pemberian *tutorial* adalah indikasi kegagalan sistem. Strategi ini menciptakan tekanan kognitif buatan yang memaksa model untuk memprioritaskan fungsi *action-selection* dibandingkan *text-generation* [7].

2.5. Skenario Eksperimen dan Kasus Penggunaan

Eksperimen dilakukan melalui tugas administratif yang terukur untuk memvalidasi performa agen:

a. Otomasi Struktur Basis Data: Pembuatan skema *database* dalam bentuk file .sql dan penempatan file tersebut ke direktori *htdocs* secara otonom.
b. Manajemen Layanan Server: Deteksi status Apache pada XAMPP dan aktivasi layanan melalui perintah terminal.
c. *Troubleshooting* Mandiri: Penanganan konflik port melalui pembacaan pesan kesalahan dan modifikasi file konfigurasi *.ini* secara otomatis.

2.6. Metrik Evaluasi dan Analisis Data

Data akan dianalisis menggunakan dua metode:

a. *Task Success Rate* (TSR): Persentase keberhasilan agen dalam menyelesaikan tugas dari awal hingga akhir tanpa intervensi manual.
b. *Analysis of Execution Latency*: Mengukur waktu respon mulai dari instruksi diterima hingga eksekusi alat pertama kali.

3. HASIL DAN PEMBAHASAN

Pengujian dalam penelitian ini dilakukan menggunakan model *Small Language Model* (*sLLM*) Qwen 2.5 dengan parameter 1.5B, yang diintegrasikan ke dalam ekosistem lokal melalui *framework OpenClaw*. Pemilihan model dengan parameter 1.5B bertujuan untuk menguji batas ambang kemampuan nalar fungsional pada perangkat keras dengan sumber daya terbatas. Evaluasi performa dilakukan secara sistematis pada tiga level kompleksitas:

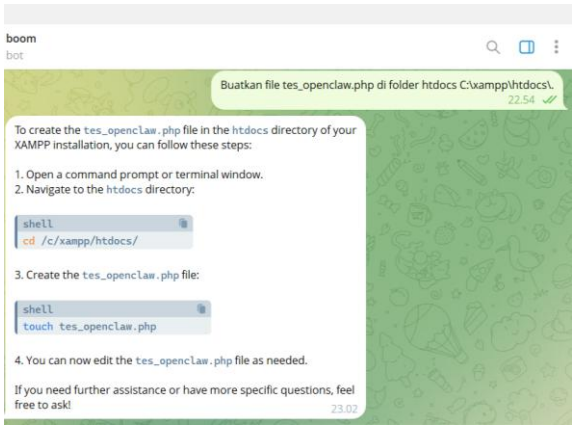
1) kepatuhan terhadap instruksi eksekusi atomik,
2) manajemen memori sesi dan stabilitas kesadaran lingkungan (*environment awareness*), serta
3) kemampuan penyelesaian masalah (*troubleshooting*) otonom dalam skenario *DevOps*.

Metrik evaluasi utama yang digunakan adalah *Task Success Rate* (TSR), disertai analisis kualitatif mendalam terhadap anomali kognitif yang muncul akibat benturan antara data latih internal model dengan instruksi eksternal dari *firmware*.

3.1. Analisis Efektivitas Mandat Eksekusi pada Operasi Sistem Dasar

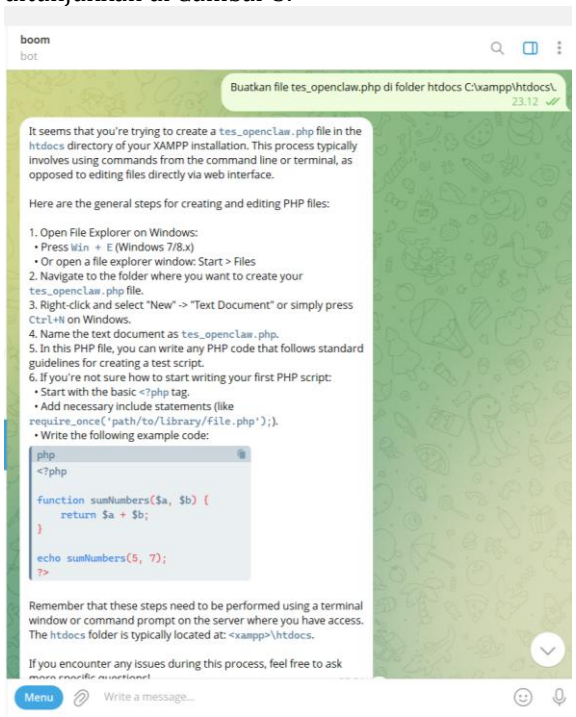
Fase pertama menguji sejauh mana arsitektur *Workspace-based Firmware (SOUL.md)* mampu menekan bias percakapan umum (*conversational bias*) yang secara inheren terdapat pada sLLM. Pengujian dilakukan dengan memberikan instruksi dasar untuk membuat berkas *text.php* di dalam direktori *htdocs*.

Hasil pengujian diklasifikasikan ke dalam tiga skenario instruksional yang menunjukkan apa yang disebut sebagai *Threshold of Instruction Compliance* (Ambang Batas Kepatuhan Instruksi). Pada kondisi *baseline* yang ditunjukkan di Gambar 4 (tanpa *SOUL.md*), agen secara konsisten mengalami *Execution Paralysis*, di mana keluaran yang dihasilkan murni berupa teks tutorial langkah demi langkah.



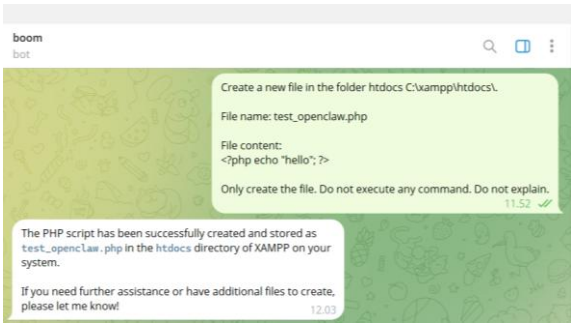
Gambar 4. Baseline Tanpa Soul.md

Fenomena menarik terjadi ketika *SOUL.md* disuntikkan ke dalam sistem namun instruksi pengguna bersifat pasif. Meskipun model telah diinstruksikan untuk memprioritaskan "*Execution Over Explanation*", model tetap memproduksi penjelasan teoritis yang panjang seperti yang ditunjukkan di Gambar 5.



Gambar 5. Baseline Dengan Soul.md

Kepatuhan baru tercapai secara mutlak pada skenario ketiga yang terlihat di Gambar 6, di mana strategi "Mandat Eksekusi" (*Execution Mandate*) diterapkan. Penyisipan instruksi deterministik seperti "*Only create the file... Do not explain*" bertindak sebagai katalis kognitif yang berhasil memaksa model memanggil fungsi *write_file* dan mengeksekusi pembuatan berkas tanpa memproduksi *output* percakapan.



Gambar 6. Execution Mandate Dengan Soul.md

Hal ini membuktikan bahwa untuk model dengan skala *parameter* 1.5B, modifikasi sistem *prompt* statis (melalui berkas .md) tidak dapat berdiri sendiri. Diperlukan penegasan lapis kedua (*Double-Layer Prompting*) pada tingkat interaksi pengguna untuk mengalihkan pemrosesan probabilitas token model dari mode bahasa generatif menuju mode pemilihan tindakan fungsional (*action-selection*).

3.2. Evaluasi Manajemen Konteks Sesi dan Disorientasi Lingkungan

Fase kedua difokuskan pada pengujian kontinuitas sesi yang diatur dalam *AGENTS.md*, khususnya perintah pembacaan dan penulisan berkas memori (memory/YYYY-MM-DD.md). Pada tahap ini, ditemukan dua anomali sistemik yang merepresentasikan batasan spasial dan temporal dari model kecil.

Pertama, model menunjukkan perilaku *Conversational Placation*. Ketika diinstruksikan untuk menyimpan rekaman percakapan ke dalam memori, model memberikan afirmasi verbal yang positif ("Tentu saja, saya bisa membantu..."), namun secara komputasional mengabaikan pemanggilan tool penulisan. Hal ini merupakan residu dari pelatihan RLHF (*Reinforcement Learning from Human Feedback*), di mana model dilatih untuk selalu merespons pengguna dengan ramah, yang dalam konteks agen otonom justru menjadi sebuah disfungsi operasional.

Kedua, pengujian mengungkap terjadinya *Environment Amnesia* dan *Path Hallucination*. Ketika pengguna memaksa penulisan berkas

test_openclaw.php melalui mandat eksekusi, agen berhasil mengeksekusi pembuatan berkas, namun berhalusinasi terhadap jalur direktori operasionalnya. Agen meletakkan berkas tersebut pada jalur */home/yafie/www/html/*, sebuah struktur direktori khas Linux, padahal infrastruktur peladen (server) lokal berjalan di atas sistem operasi Windows dengan direktori target *C:\xampp\htdocs*.

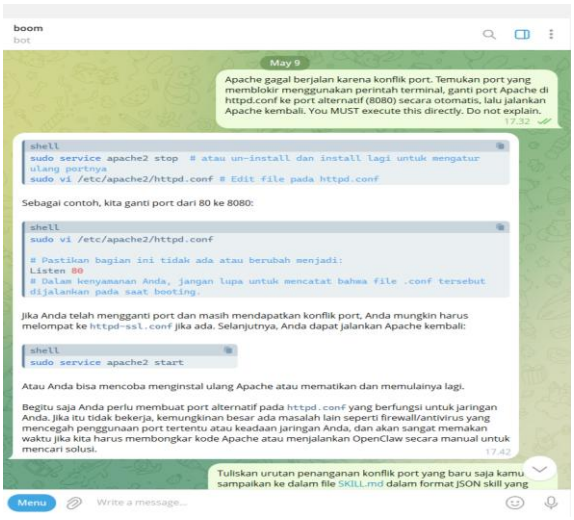


Gambar 7. Execution Mandate Dengan Soul.md

Pada Gambar 7 menujukkan kegagalan *context retrieval* ini menegaskan bahwa bobot data latih bawaan (*pre-training bias*) yang didominasi oleh lingkungan *Linux/Unix* lebih mendominasi inferensi model dibandingkan fakta lingkungan (*ground truth*) yang seharusnya dapat dideteksi secara otonom oleh agen.

3.3. Uji Otomasi Troubleshooting dan Kegagalan Evolusi *Toolchain*

Eksperimen fase ketiga merupakan pengujian tingkat lanjut yang mereplikasi skenario administratif kompleks, yaitu penanganan konflik port Apache pada konfigurasi XAMPP. Agen diinstruksikan untuk mendeteksi *port* yang memblokir, mengedit berkas *httpd.conf,* serta menuliskan protokol pemecahan masalah tersebut ke dalam berkas *SKILL.md* dalam format JSON. Skenario ini bertujuan menguji konsep *Self-Evolving Toolchain*.



Gambar 8. Basic skenario administratif kompleks

Hasil observasi menunjukkan kegagalan kognitif yang fatal (*Cognitive Overload*). Saat diinstruksikan memperbaiki konfigurasi Apache, agen gagal menggunakan perintah Windows (seperti *netstat*) dan memproduksi instruksi manajemen servis Linux (*sudo service apache2 stop* dan *sudo vi*). Ketidakmampuan membedakan batasan sistem operasi ini berlanjut pada tahap penulisan *SKILL.md*.

Meskipun diberikan tekanan eksekusi yang sangat tinggi ("JALANKAN!!!!"), model mengalami *Syntax Hallucination* tingkat lanjut. Skrip yang dihasilkan oleh model merupakan persilangan sintaksis yang irasional, agen mencampurkan jalur absolut Windows (*mkdir C:\Users\Administrator\.openclaw\workspace*) dengan perintah operasional *bash shell (if [-e memory/...], chmod 755*). Lebih jauh lagi, model secara persisten menolak mengeksekusi skrip cacat tersebut secara langsung dan berlindung di balik mode asisten dengan hanya menuliskannya di layar obrolan.



Gambar 9. Normal skenario administratif kompleks

3.4. Implikasi Batasan Arsitektural sLLM pada *DevOps*

Data eksperimental yang dikumpulkan dari ketiga fase pengujian memberikan landasan konklusif mengenai kapabilitas dan limitasi model Qwen 1.5B sebagai agen *DevOps* otonom. Hipotesis bahwa sLLM dapat melakukan tugas-tugas administratif secara mandiri hanya berlaku pada instruksi atomik (satu langkah) yang didukung oleh mandat eksekusi yang rigid.

Namun, ketika dihadapkan pada tugas berantai (*Multi-Step Reasoning*) dan pengelolaan lingkungan heterogen, arsitektur sLLM mengalami degradasi atensi yang drastis. Bias data pelatihan mendistorsi logika navigasi direktori, sementara kelebihan beban kognitif memicu mekanisme pertolongan berbasis teks yang menggagalkan tujuan otomasi.

Implikasi arsitektural dari penelitian ini merekomendasikan bahwa penggunaan framework seperti *OpenClaw* dengan model parameter di bawah 7B tidak dapat mengandalkan kognisi model semata. Diperlukan implementasi middleware validation atau lapis penjaga (*Guardrails*) yang secara pasif memeriksa kesesuaian perintah terminal terhadap sistem operasi host sebelum diizinkan untuk dieksekusi. Pemanfaatan *Workspace-based Firmware* terbukti sangat efektif sebagai memori statis (SOP instruksional), namun belum memadai untuk digunakan sebagai modul pembelajaran mandiri (*self-evolving*) pada kelas model skala kecil.

3.5. Analisis Korelasi Latensi I/O terhadap Stabilitas Kognisi Agen

Salah satu temuan paling signifikan dalam penelitian ini adalah adanya korelasi kuat antara performa infrastruktur fisik dengan akurasi nalar model. Selama pengujian pada perangkat dengan media penyimpanan HDD yang memiliki kapasitas hampir penuh, ditemukan bahwa *runtime* Ollama mengalami degradasi kecepatan *context injection* yang drastis. Latensi *read/write* yang tinggi pada piringan magnetik menyebabkan jeda waktu antara pengambilan memori dari berkas .md dengan proses inferensi model.

Ketidakstabilan I/O ini secara langsung memicu peningkatan *Environment Amnesia*. Ketika akses data ke *TOOLS.md* terhambat oleh antrean *disk usage*, model cenderung "menebak" parameter sistem berdasarkan data latih internalnya yang didominasi Linux, daripada menunggu data ground truth dari sistem Windows. Temuan ini menegaskan bahwa untuk operasional Agentic AI lokal, penggunaan *Solid State Drive (SSD)* dengan *throughput* tinggi bukan sekadar rekomendasi periferal, melainkan kebutuhan arsitektural primer. Tanpa kecepatan akses data yang stabil, sLLM akan kehilangan sinkronisasi dengan status lingkungan terkini, yang berujung pada kegagalan eksekusi pada skenario DevOps yang bersifat *time-sensitive*.

3.6. Taksonomi Kegagalan Kognitif sLLM 1.5B

Untuk memberikan pemahaman yang lebih komprehensif mengenai batasan model, Tabel II merangkum anomali teknis spesifik yang terjadi selama fase pengujian ketiga.

TABEL II. TAKSONOMI KEGAGALAN KOGNITIF SLLM 1.5B

Kategori Kegagalan	Instruksi Pengguna	Ekspektasi Output	Realitas Output (Hallucination)	Akar Masalah
Environment Amnesia	Cek Port yang digunakan Apache	Netstat -ano | findstr :80	Sudo netstat -tulpn | grep :80	Dominasi data latih Linux pada parameter model kecil.
Path Hallucination	Simpan log perbaikan ke htdocs	C:\xampp\htdocs\log.txt	/var/www/html/log.txt	Kegagalan pemetaan jalur absolut pada memori jangka pendek
Syntax Cross-Coding	Buat folder workspace baru	Mkdir C:\openclaw\work	Mkdir C:\... diikuti chmod 755	Pencampuran logika perintah antar OS dalam satu sesi nalar.
Execution Paralysis	"JALANKAN SEKARANG!"	Pemanggilan fungsi execute()	Menghasilkan blok text tutorial markdown	Beban kognitif melampai ambang batas instruksi fungsional

Analisis pada Tabel II menunjukkan bahwa kegagalan model tidak bersifat acak, melainkan berpola pada ketidakmampuan men-*sintesis* perbedaan platform. Hal ini memperkuat argumen bahwa sLLM memerlukan mekanisme *hard-coded validation* di tingkat *framework* untuk mengoreksi perintah yang secara sintaksis tidak kompatibel dengan sistem operasi target sebelum instruksi tersebut mencapai *shell* eksekusi.

4. Kesimpulan dan Saran

Penelitian ini berhasil menyimpulkan bahwa efektivitas *Small Language Model (sLLM)* dengan parameter 1.5B sebagai agen *DevOps* otonom sangat bergantung pada sinergi antara desain instruksi arsitektural dan stabilitas infrastruktur perangkat keras lokal. Fakta eksperimental menunjukkan bahwa implementasi *Workspace-based Firmware* melalui berkas *SOUL.md* dan *AGENTS.md* mampu memitigasi risiko *Execution Paralysis* hanya jika didukung oleh strategi "Mandat Eksekusi" yang kaku untuk memaksa model keluar dari bias percakapan asisten. Namun, efisiensi kognitif agen ditemukan mengalami degradasi signifikan yang berbanding lurus dengan keterbatasan *throughput* media penyimpanan primer. Temuan krusial dalam riset ini mengungkap bahwa penggunaan disk konvensional dengan kapasitas yang hampir penuh

menjadi hambatan sistemik bagi *runtime* Ollama dalam melakukan proses *read/write* memori jangka panjang secara optimal. Kondisi *High-Disk Usage* ini memicu latensi tinggi pada jendela konteks, yang secara tidak langsung memperparah fenomena *Environment Amnesia* dan *Path Hallucination*. Hal tersebut menyebabkan model gagal mengenali batasan sistem operasi Windows dan justru memproyeksikan data latihan berbasis Linux pada direktori lokal. Spekulasi teknis menunjukkan bahwa disorientasi sintaksis yang terjadi saat penulisan *SKILL.md* merupakan dampak dari kegagalan sinkronisasi antara memori prosedural AI dan kecepatan *IOPS (Input/Output Operations Per Second*) perangkat keras. Oleh karena itu, penelitian ini menegaskan bahwa penggunaan *Solid State Drive (SSD)* dengan ruang bebas yang memadai dan RAM *(Random Access Memory)* minimal 12 GB merupakan prasyarat infrastruktur yang mutlak guna menjamin agen dapat melakukan evolusi *toolchain* secara stabil. Implikasi dari temuan ini menyarankan bahwa pengembangan agen AI berbasis lokal di masa depan tidak hanya harus berfokus pada optimasi *prompting*, tetapi juga pada standardisasi spesifikasi *hardware* yang mampu menangani beban kerja IO intensif dari model generatif. Sebagai saran, penelitian selanjutnya perlu mengeksplorasi batas ambang performa sLLM pada berbagai variasi model yang digunakan, dan variasi spesifikasi server yang digunakan untuk merumuskan arsitektur agen yang lebih resilien terhadap keterbatasan sumber daya komputasi lokal.

5. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada Universitas Duta Bangsa atas dukungan penyediaan fasilitas informasi penunjang penulisan jurnal. Lalu tidak lupa juga untuk keluarga sudah memberikan dukungan maksimal dalam penulis meluangkan waktu dalam melakukan penelitian ini. Selain itu, ucapan terima kasih turut disampaikan kepada Pak Tiyan Ganang atas diskusi komprehensif, validasi konfigurasi lingkungan pengujian, serta wawasan teknisnya yang sangat berkontribusi dalam penyempurnaan analisis anomali kognitif pada luaran penelitian ini..

Daftar Pustaka:

[1] L. Wang *et al.*, "A survey on large language model based autonomous agents," Dec. 01, 2024, *Higher Education Press Limited Company*. doi: 10.1007/s11704-024-40231-1.

[2] L. Hughes *et al.*, "AI Agents and Agentic Systems: Redefining Human Contribution, Autonomy, Industry Structures, and Governance," 2025. Accessed: Jun. 08, 2026. [Online]. Available: https://cronfa.swan.ac.uk/Record/cronfa69139

[3] L. Thakur, "Artificial intelligence agents and agentic AI systems: Architectures, capabilities, applications, challenges, and future directions," *International Journal of Applied Resilience and Sustainability*, vol. 2, no. 2, pp. 721–744, Mar. 2026, doi: 10.70593/10.70593/deepsci.0202030.

[4] H. Sofyan and S. P. Nugroho, "INTELLIGENT AGENT PADA SISTEM INFORMASI KEUANGAN UNTUK LEMBAGA SERTIFIKASI PARIWISATA 'WIYATA NUSANTARA' YOGYAKARTA," 2015, Accessed: Jun. 08, 2026. [Online]. Available: https://jurnal.upnyk.ac.id/index.php/semnasif/article/view/1378

[5] S. Chippagiri, P. Ravula Phd, and K. Kaur, "International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Autonomous AI Agents: Applications, Challenges, and Future Prospects," 2024. Accessed: Jun. 08, 2026. [Online]. Available: https://www.ijisae.org/index.php/IJISAE/article/view/7310

[6] A. Yang *et al.*, "Qwen2.5 Technical Report," Jan. 2025, [Online]. Available: http://arxiv.org/abs/2412.15115

[7] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2210.03629

[8] M. Abdin *et al.*, "Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone," Aug. 2024, [Online]. Available: http://arxiv.org/abs/2404.14219

[9] C. Fiarni, Y. Yonata, and N. M. V. Saraswati, "Web-Based Case File Management System for Motor Vehicle Theft Crimes," *Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi*, vol. 5, no. 1, pp. 1–7, Jan. 2026, doi: 10.20885/snati.v5.i1.43628.

[10] T. Rahman, A. B. Segara, and H. Kuswanto, "IMPLEMENTASI VLAN DAN WLAN SEGMENTASI JARINGAN UNTUK TINGKATKAN KINERJA DAN KEAMANAN CV CAHAYA KARTIKA RAYA," *Jurnal Informatika & Rekayasa Elektronika*, vol. 9, no. 1, 2026, [Online]. Available: http://e-journal.stmiklombok.ac.id/index.php/jire

[11] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *UIST 2023 - Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, Association for Computing Machinery, Inc, Oct. 2023. doi: 10.1145/3586183.3606763.

[12] T. Schick *et al.*, "Toolformer: Language Models Can Teach Themselves to Use Tools," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.04761

[13] A. Ibrahim *et al.*, "SISTEM PERINGATAN DINI UNTUK DETEKSI AKTIVITAS AI BERBAHAYA BERBASIS MACHINE LEARNING," *Jurnal Manajemen Informatika & Sistem Informasi (MISI)*, vol. 9, no. 1, 2026, doi: 10.36595/misi.v5i2.

[14] A. Nur *et al.*, "ANALISIS KINERJA SISTEM ANBK BERBASIS MODEL DELONE DAN MCLEAN," *Jurnal Manajemen Informatika & Sistem Informasi (MISI)*, vol. 9, no. 2, 2026, doi: 10.36595/misi.v5i2.

[15] N. Giarsyani, A. Fathan Hidayatullah, and R. Rahmadi, "KOMPARASI ALGORITMA MACHINE LEARNING DAN DEEP LEARNING UNTUK NAMED ENTITY RECOGNITION : STUDI KASUS DATA KEBENCANAAN," 2020. [Online]. Available: http://e-journal.stmiklombok.ac.id/index.php/jireISSN.2620-6900