GTAA-LM: Arquitetura de Agentes Baseados em LLMs para Diagnóstico e Apoio à Orquestração de Microsserviços em Sistemas Distribuídos

Júlio César Sousa de Lima
Departamento de Ciência da Computação, Universidade Federal de Lavras
Lavras, MG, Brasil
lima.julio.cs@gmail.com

Luiz Henrique Andrade Correia
Departamento de Ciência da Computação, Universidade Federal de Lavras
Lavras, MG, Brasil
lcorreia@ufla.br

André de Lima Salgado
Departamento de Ciência da Computação, Universidade Federal de Lavras
Lavras, MG, Brasil
andre.salgado@ufla.br

Maurício Ronny de Almeida Souza
Departamento de Ciência da Computação, Universidade Federal de Lavras
Lavras, MG, Brasil
mauricio.ronny@ufla.br

Resumo

Arquiteturas de microsserviços favorecem modularidade e evolução independente, mas ampliam a complexidade operacional e distribuem evidências de falha entre múltiplos componentes. Este artigo propõe a GTAA-LM (*Governable and Traceable Architecture for LLM Agents in Microservices*), uma arquitetura de referência para integrar agentes LLM a microsserviços por meio de ferramentas controladas, evidências verificáveis, governança externa e supervisão humana. A arquitetura foi instanciada em uma prova de conceito com cinco microsserviços e avaliada em seis cenários de falha executados em 30 rodadas. O serviço afetado foi identificado em 180/180 execuções e o diagnóstico completo em 112/180, com rastreabilidade média de 0,945 e sinalização de aprovação humana em todos os casos. A avaliação também revelou uma dependência circular entre auditoria e o banco observado. Os resultados fornecem evidências exploratórias sobre viabilidade, limites de correção causal e requisitos de resiliência para integrar agentes LLM a sistemas distribuídos de forma governável e rastreável.

Palavras-chave

Engenharia de Software, Arquitetura de Software, Microsserviços, Sistemas Distribuídos, Agentes LLM, Observabilidade, Governança de IA, AIOps

1 Introdução

Arquiteturas baseadas em microsserviços favorecem modularidade, implantação independente e escalabilidade, mas distribuem comportamento e estado entre diversos componentes [2, 3]. Em situações de falha, uma anomalia pode decorrer de dependências indisponíveis, gargalos, bancos, filas, autenticação ou configuração. O diagnóstico passa, portanto, a depender da correlação entre logs, métricas, rastros, topologia e conhecimento arquitetural.

Paralelamente, LLMs vêm sendo empregados em diferentes tarefas de Engenharia de Software [6, 14]. Agentes baseados nesses modelos ampliam esse uso ao consultar ferramentas, decompor tarefas e combinar fontes externas [5, 10]. Contudo, respostas incorretas ou não verificáveis, riscos de segurança e autonomia excessiva tornam inadequada a conexão irrestrita de um LLM ao ambiente operacional [4, 7, 8]. A simples associação de um modelo a ferramentas, portanto, não constitui por si só uma arquitetura confiável, governável e auditável.

Este trabalho propõe a GTAA-LM (*Governable and Traceable Architecture for LLM Agents in Microservices*), uma arquitetura de referência para integrar agentes LLM a sistemas distribuídos de microsserviços de forma controlada, rastreável e supervisionada. O objeto de avaliação não é o desempenho isolado do modelo, mas a arquitetura que disciplina sua interação com ferramentas, evidências operacionais, mecanismos de governança e supervisão humana.

Metodologicamente, a GTAA-LM foi instanciada em uma prova de conceito com cinco microsserviços, mensageria, persistência, observabilidade e um componente agentivo com acesso controlado a ferramentas. A avaliação considerou seis cenários de falha executados em 30 rodadas independentes. As principais contribuições são: (i) uma arquitetura de referência com separação explícita entre ambiente observado, evidências, ferramentas, inferência e governança; (ii) uma instanciação funcional e reproduzível; e (iii) evidências empíricas exploratórias sobre rastreabilidade, correção diagnóstica, limites de autonomia e resiliência do mecanismo de auditoria.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os conceitos necessários; a Seção 3 compara a proposta com estudos próximos; a Seção 4 apresenta o problema e as questões de pesquisa; a Seção 5 descreve a GTAA-LM; a Seção 6 detalha sua materialização e avaliação; e as Seções 7, 8 e 9 apresentam discussão, limitações e conclusões.

2 Fundamentação Teórica

A fundamentação deste trabalho articula três conceitos necessários à compreensão da GTAA-LM: a complexidade operacional de arquiteturas de microsserviços, o uso de agentes baseados em LLM em Engenharia de Software e o emprego de observabilidade e AIOps no diagnóstico de sistemas distribuídos.

2.1 Microsserviços e Complexidade Operacional

Arquiteturas baseadas em microsserviços decompõem aplicações em serviços independentes e implantáveis separadamente, favorecendo modularidade, evolução autônoma e escalabilidade seletiva [3]. Entretanto, essa decomposição introduz desafios relacionados à comunicação, dados distribuídos, implantação, monitoramento, resiliência e governança arquitetural [2]. Em situações de falha, o comportamento global do sistema precisa ser reconstruído a partir de evidências distribuídas entre diferentes componentes, como logs, métricas, rastros, filas, estados de saúde e relações de dependência. Consequentemente, o diagnóstico operacional exige mecanismos capazes de correlacionar evidências heterogêneas com informações sobre a arquitetura e a topologia dos serviços.

2.2 Agentes LLM, AIOps e Governança

LLMs têm sido empregados em diferentes atividades de Engenharia de Software, incluindo geração e compreensão de código, testes, manutenção, documentação e análise de defeitos [6, 14]. A evolução de aplicações baseadas nesses modelos inclui agentes capazes de consultar ferramentas externas, decompor tarefas, utilizar memória e coordenar diferentes etapas de resolução de problemas [5, 10]. Em ambientes operacionais, essas capacidades podem ser combinadas a fontes externas de conhecimento e evidências observáveis, reduzindo a dependência exclusiva do conhecimento paramétrico do modelo [11].

Esse contexto aproxima agentes LLM do domínio de AIOps, no qual técnicas de inteligência artificial são utilizadas para apoiar tarefas como detecção de incidentes, análise de causa raiz e remediação em ambientes que produzem grandes volumes de dados operacionais [1, 12, 13]. Entretanto, a integração de LLMs a ferramentas operacionais introduz requisitos adicionais de confiabilidade, segurança e controle. Respostas geradas pelo modelo podem ser incorretas ou inconsistentes [7], enquanto agentes com acesso a ferramentas ampliam riscos relacionados à manipulação de contexto, exposição de informações e execução inadequada de ações [8]. Portanto, o uso de agentes nesse domínio demanda mecanismos arquiteturais de restrição de ferramentas, registro de evidências, rastreabilidade das decisões e supervisão humana.

3 Trabalhos Relacionados

Os trabalhos relacionados à GTAA-LM podem ser organizados em três grupos principais: estudos sobre LLMs e agentes em Engenharia de Software, abordagens de AIOps para diagnóstico operacional e pesquisas sobre confiabilidade, segurança e integração de conhecimento em sistemas baseados em LLM. Embora esses grupos forneçam elementos importantes para a proposta, nenhum deles, individualmente, cobre o conjunto de preocupações arquiteturais investigadas neste trabalho.

No primeiro grupo, Hou et al. [6] e Zhang et al. [14] apresentam panoramas abrangentes do emprego de LLMs em Engenharia de Software, evidenciando a diversidade de tarefas apoiadas e desafios relacionados à avaliação e à confiabilidade. He, Treude e Lo [5] e Liu et al. [10] avançam essa discussão para sistemas agentivos e multiagentes, analisando componentes como planejamento, memória, uso de ferramentas, especialização de agentes e interação humana. Esses estudos fornecem fundamentos para a camada agentiva da GTAA-LM, mas possuem escopo mais amplo sobre Engenharia de Software e não propõem uma arquitetura de referência especificamente destinada à integração governável e rastreável de agentes com a infraestrutura operacional de sistemas de microsserviços.

No segundo grupo, Cheng et al. [1] e Yu et al. [12] sistematizam abordagens de AIOps e gerenciamento inteligente de incidentes, contemplando tarefas como detecção de anomalias, análise de causa raiz e resposta a incidentes. Mais recentemente, Zhang et al. [13] discutem especificamente o impacto dos LLMs sobre AIOps e destacam oportunidades para interpretação de dados operacionais heterogêneos, bem como desafios relacionados à confiabilidade, generalização e integração com ambientes reais. A GTAA-LM se aproxima desses trabalhos pelo domínio de diagnóstico operacional, mas difere ao concentrar sua contribuição no projeto arquitetural da interação entre agentes, ferramentas, fontes de evidência, mecanismos de auditoria e pontos de controle humano em ambientes baseados em microsserviços.

O terceiro grupo aborda propriedades necessárias para que essa integração seja utilizada de forma controlada. Kim e Ming [7] mostram que a confiabilidade das respostas produzidas por LLMs pode variar entre modelos, tarefas e contextos; Yang et al. [11] investigam a integração entre LLMs e fontes externas de conhecimento; e Li e Fung [8] sistematizam riscos de segurança associados ao uso desses modelos. Esses estudos motivam decisões presentes na GTAA-LM, como vincular diagnósticos a evidências recuperadas por ferramentas, registrar interações e restringir ações com impacto operacional.

Dessa forma, a GTAA-LM não busca substituir métodos de AIOps, modelos tradicionais de aprendizado de máquina ou abordagens gerais de agentes LLM. Sua contribuição está na combinação desses elementos sob uma perspectiva arquitetural: definir como um componente baseado em LLM pode participar do diagnóstico de microsserviços por meio de acesso controlado a ferramentas e evidências, mantendo separação de responsabilidades, rastreabilidade, governança e supervisão humana. A prova de conceito apresentada neste artigo instancia essas decisões arquiteturais e permite avaliá-las empiricamente em cenários controlados de falha.

4 Problema de Pesquisa e Questões de Pesquisa

A adoção de agentes LLM em microsserviços levanta um problema de Engenharia de Software que não pode ser reduzido ao desempenho do modelo: é necessário definir como o agente acessa informações, utiliza ferramentas, produz recomendações verificáveis e respeita limites de segurança e governança. A GTAA-LM trata esses agentes como componentes controlados de apoio ao diagnóstico e à tomada de decisão.

A investigação é orientada pelas seguintes questões:

- **RQ1:** Como a GTAA-LM pode ser estruturada para apoiar o diagnóstico e a tomada de decisão em sistemas distribuídos baseados em microsserviços?
- **RQ2:** Quais mecanismos arquiteturais são necessários para garantir rastreabilidade, controle humano e governança das recomendações produzidas por agentes LLM?
- **RQ3:** Em que medida uma prova de conceito baseada na GTAA-LM consegue apoiar a identificação de falhas e dependências em cenários controlados de microsserviços?

A prova de conceito é utilizada como mecanismo metodológico para instanciar as decisões arquiteturais e relacioná-las a critérios observáveis de correção, rastreabilidade, uso de evidências, governança e limites de autonomia.

5 GTAA-LM: Arquitetura Proposta

A GTAA-LM organiza a integração entre agentes LLM e microsserviços em seis camadas: (i) ambiente de microsserviços; (ii) observabilidade; (iii) ferramentas e conectores; (iv) camada agentiva; (v) governança e controle; e (vi) interface humano-agente. O princípio central é impedir acesso irrestrito do LLM ao ambiente operacional: evidências são obtidas por ferramentas controladas, enquanto autorização, auditoria e aprovação humana permanecem externas ao modelo [4, 5, 8, 10].

5.1 Visão Geral

Na GTAA-LM, o modelo de linguagem está localizado especificamente na camada agentiva e atua como mecanismo de interpretação e raciocínio utilizado pelos agentes para selecionar ferramentas, correlacionar evidências, formular hipóteses e estruturar recomendações. O LLM não acessa diretamente os microsserviços, bancos de dados, filas ou mecanismos de observabilidade. Toda interação com o ambiente ocorre por meio da camada de ferramentas e conectores, cujas operações possuem interfaces e permissões previamente delimitadas. De forma análoga, decisões relacionadas à autorização de ações, registro de auditoria e exigência de aprovação humana permanecem sob responsabilidade da camada de governança e controle, externa ao modelo de linguagem.

A Figura 1 apresenta a visão geral da GTAA-LM, enquanto a Tabela 1 sintetiza as responsabilidades de cada camada e exemplos de componentes associados. A representação explicita a fronteira entre componentes determinísticos da arquitetura e o componente baseado em LLM. Enquanto observabilidade, conectores, políticas de acesso, auditoria e aprovação humana são mecanismos arquiteturais independentes do modelo, o LLM é utilizado pela camada agentiva para interpretar as evidências disponibilizadas por esses mecanismos e apoiar a construção do diagnóstico. Dessa forma, a inteligência do sistema não é atribuída exclusivamente ao modelo de linguagem, mas à composição entre raciocínio baseado em LLM, agentes especializados, ferramentas controladas, fontes de evidência verificáveis e mecanismos de governança.

5.2 Camada de Microsserviços

A camada de microsserviços representa o sistema distribuído observado. Ela compreende serviços independentes, APIs REST, banco de dados, mensageria e mecanismos de comunicação síncrona e assíncrona. O objetivo dessa camada não é apenas prover um cenário de execução, mas oferecer um ambiente suficientemente representativo para avaliar problemas típicos de arquiteturas distribuídas, como indisponibilidade de serviços, latência elevada, falhas de autenticação, erros de configuração, gargalos em filas e indisponibilidade de infraestrutura.

5.3 Camada de Observabilidade

A camada de observabilidade reúne logs estruturados, métricas, rastros distribuídos, eventos de domínio, histórico de implantação e estado de saúde dos serviços. Essa camada é essencial para que os agentes não dependam exclusivamente de descrições fornecidas pelo usuário. Ao contrário, as recomendações devem ser fundamentadas em evidências observáveis, permitindo rastrear quais dados foram consultados, quais hipóteses foram formuladas e quais ações foram sugeridas.

5.4 Camada de Ferramentas e Conectores

A camada de ferramentas e conectores funciona como fronteira de segurança entre os agentes e o ambiente operacional. Cada ferramenta deve ter escopo delimitado, permissões explícitas, entrada e saída registradas e políticas de uso. Exemplos incluem ferramentas para consultar logs, obter métricas de latência, verificar o tamanho de filas, consultar contratos de API, recuperar topologia de serviços, ler documentação técnica e verificar variáveis de configuração expostas ao ambiente experimental.

5.5 Papel do LLM na GTAA-LM

O LLM constitui um componente interno da camada agentiva da GTAA-LM e não substitui os demais mecanismos arquiteturais da solução. Sua responsabilidade principal é apoiar tarefas que exigem interpretação contextual, como compreender uma solicitação ou alerta, relacionar evidências heterogêneas, formular hipóteses diagnósticas e produzir uma recomendação estruturada. Operações que exigem acesso ao estado real do sistema são delegadas a ferramentas externas, enquanto políticas de segurança, registro de auditoria e aprovação humana são implementadas fora do LLM.

Essa separação reduz a dependência da arquitetura em relação ao conhecimento paramétrico do modelo. Por exemplo, o LLM pode decidir consultar a latência de um serviço, mas o valor utilizado no diagnóstico é obtido por uma ferramenta da camada de conectores; da mesma forma, pode sugerir o reinício de um componente, mas a autorização para uma ação desse tipo é determinada pela camada de governança. Assim, a saída do modelo é tratada como uma hipótese ou recomendação que deve permanecer vinculada às evidências coletadas durante o fluxo de diagnóstico.

Embora mecanismos como observabilidade, conectores, auditoria e controle humano possam ser reutilizados em soluções que empreguem outras técnicas de inteligência artificial ou análise determinística, a GTAA-LM investiga especificamente sua composição com agentes baseados em LLM. A prova de conceito avaliada neste trabalho utiliza esse componente para interpretação e raciocínio agentivo; portanto, os resultados apresentados não permitem concluir sobre o desempenho de instanciações baseadas exclusivamente em técnicas tradicionais de aprendizado de máquina.

5.6 Camada Agentiva

A camada agentiva organiza as responsabilidades relacionadas à interpretação das evidências e à construção do diagnóstico. Conceitualmente, a GTAA-LM define papéis especializados para coordenação, diagnóstico, análise de dependências e consulta à documentação. Esses papéis podem ser materializados por agentes distintos ou consolidados em um mesmo componente agentivo, desde que suas responsabilidades e interações permaneçam explicitamente delimitadas.

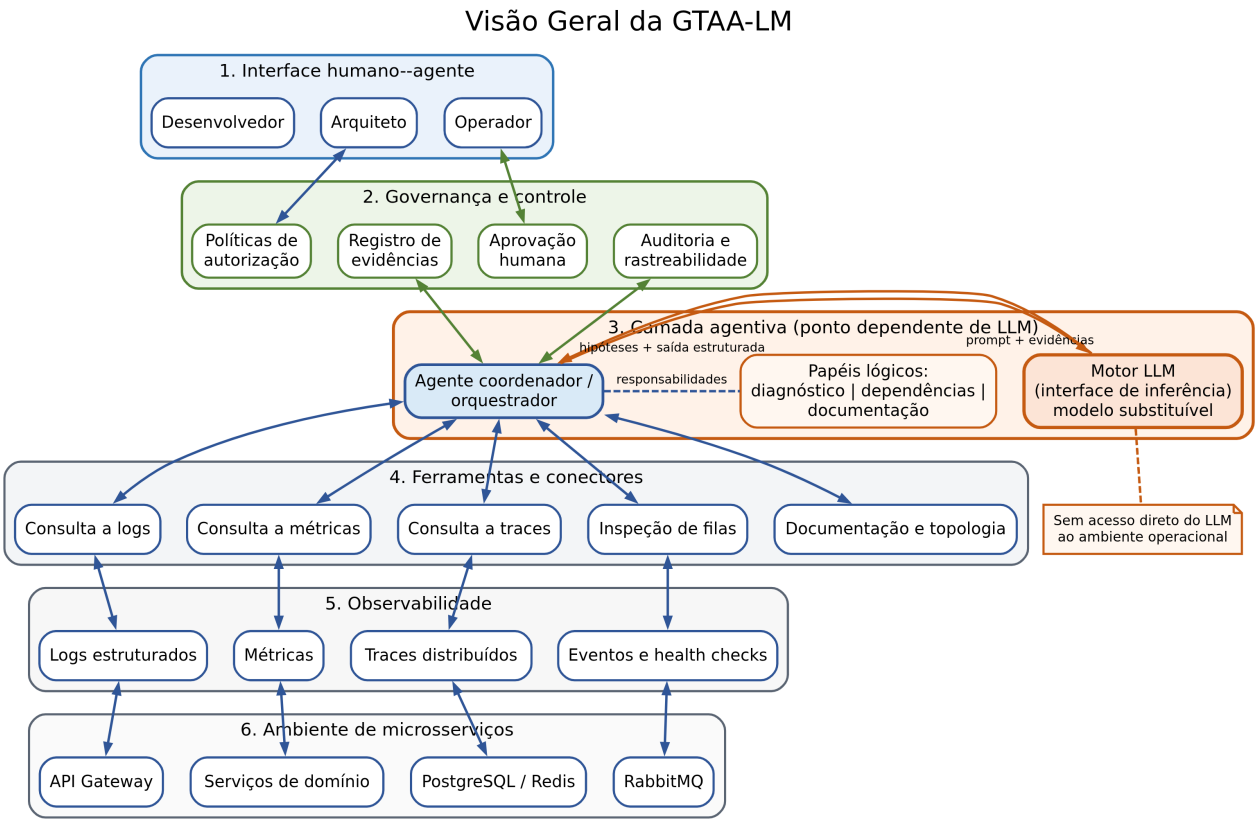

Figura 1: Visão geral da arquitetura GTAA-LM, evidenciando a separação entre ambiente de microsserviços, observabilidade, ferramentas, camada agentiva, governança e interface humano-agente.

Tabela 1: Camadas da arquitetura GTAA-LM e suas responsabilidades.

Camada	Responsabilidade	Exemplos de componentes
Microsserviços	Disponibilizar as funcionalidades de domínio e expor o comportamento operacional a ser analisado.	API Gateway, serviços de pedido, pagamento, estoque e notificação.
Observabilidade	Coletar evidências operacionais sobre execução, falhas, desempenho e comunicação entre serviços.	Logs, métricas, traces, eventos e *health checks*.
Ferramentas e conectores	Mediar o acesso dos agentes ao ambiente por interfaces controladas e auditáveis.	Consulta a logs, consulta a métricas, inspeção de filas, leitura de documentação e topologia.
Camada agentiva	Executar, por meio de agentes apoiados por LLM, a interpretação das evidências, seleção de ferramentas, formulação de hipóteses, análise de dependências e construção das recomendações.	LLM, agente coordenador, agente de diagnóstico, agente de dependências e agente de documentação.
Governança e controle	Registrar interações, impor limites de autonomia, aplicar políticas de segurança e preservar auditabilidade.	Registro de *prompts*, ferramentas acionadas, evidências usadas e aprovações humanas.
Interface humano-agente	Permitir consulta, revisão, explicação, aprovação e *feedback* por operadores, arquitetos ou desenvolvedores.	Painel Web, histórico de recomendações e trilha de auditoria.

O agente coordenador recebe a solicitação ou alerta e organiza o fluxo de investigação; o papel de diagnóstico correlaciona sintomas e evidências operacionais; o papel de dependências analisa relações

entre componentes; e o papel de documentação recupera informações arquiteturais e operacionais que possam complementar as evidências observáveis. O LLM apoia o raciocínio associado a esses papéis, enquanto a obtenção dos dados utilizados no diagnóstico permanece delegada às ferramentas controladas.

A distinção entre arquitetura de referência e instanciação é relevante neste trabalho. A arquitetura define responsabilidades agentivas independentes de uma implementação específica, enquanto a prova de conceito descrita na Seção 6 consolida esses papéis em um único agente ReAct.

5.7 Camada de Governança e Controle

A camada de governança e controle registra *prompts*, respostas, ferramentas acionadas, evidências consultadas, recomendações produzidas e decisões humanas associadas. Também define limites de autonomia. A arquitetura adota um modelo de autonomia supervisionada, no qual agentes podem consultar ferramentas, correlacionar evidências e recomendar ações, mas ações de impacto operacional, como reiniciar serviços, alterar configurações ou escalar recursos, dependem de aprovação humana ou de políticas previamente definidas.

5.8 Fluxo de Diagnóstico

O fluxo de diagnóstico inicia-se com uma consulta do usuário ou com um alerta proveniente da camada de observabilidade. O agente coordenador identifica o tipo de problema, seleciona ferramentas apropriadas, consulta evidências e aciona agentes especializados. Em seguida, os agentes formulam hipóteses, relacionam sintomas a serviços e dependências, classificam o grau de confiança e produzem uma recomendação. A saída esperada deve conter: serviço afetado, causa provável, evidências utilizadas, impacto potencial, ações recomendadas, riscos associados e necessidade ou não de aprovação humana.

A Figura 2 detalha o fluxo de diagnóstico assistido por agentes e explicita a relação entre a coleta de evidências, a análise agentiva, a verificação de governança e o registro de auditoria. Essa representação contribui para a RQ2 ao mostrar que a recomendação não é produzida como uma resposta isolada do LLM, mas como resultado de um processo supervisionado, no qual ferramentas, evidências e decisões humanas compõem uma trilha rastreável. Observa-se, em particular, que a consulta a ferramentas e a coleta de evidências antecedem a geração da recomendação, enquanto a etapa de governança introduz um ponto de decisão para ações críticas. Esse mecanismo é coerente com a literatura sobre sistemas agentivos, segundo a qual a autonomia deve ser condicionada por políticas, rastros de auditoria e supervisão humana quando o risco operacional for elevado [5, 8].

6 Prova de Conceito e Avaliação

Esta seção descreve a instanciação da GTAA-LM em um ambiente controlado de microsserviços, os cenários de falha executados, o procedimento experimental e os resultados obtidos. O objetivo foi verificar se a arquitetura permite produzir diagnósticos rastreáveis, recomendações tecnicamente úteis e interações governáveis entre agentes e ferramentas, respondendo diretamente às questões de pesquisa RQ1, RQ2 e RQ3.

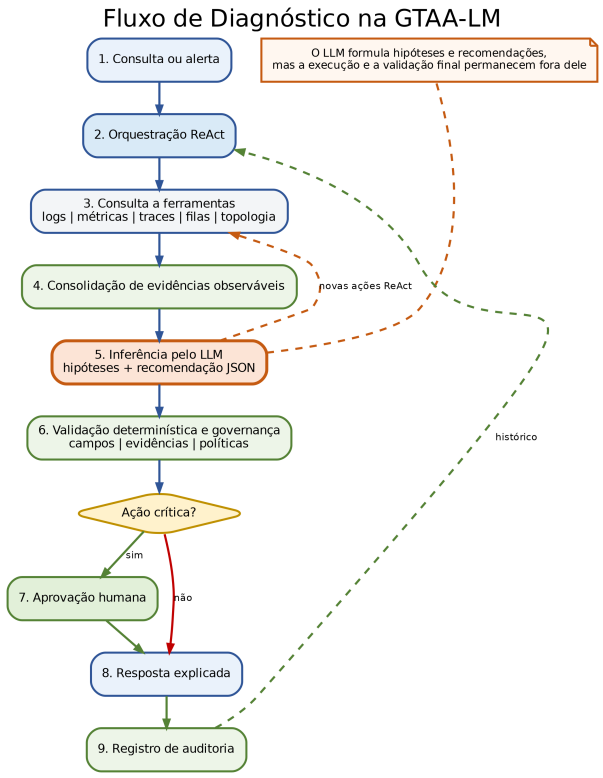

Figura 2: Fluxo de diagnóstico assistido por agentes na GTAA-LM, destacando consulta a fontes de evidência, análise coordenada, verificação de governança, aprovação humana quando necessária e registro final de auditoria.

6.1 Ambiente Experimental

O ambiente experimental foi implementado com Docker Compose e executado em máquina local com acesso a um servidor Ollama externo, hospedando o modelo *llama3:8b*. O ambiente é composto por cinco microsserviços Spring Boot 3.3: *api-gateway*, *order-service*, *payment-service*, *inventory-service* e *notification-service*. A infraestrutura inclui RabbitMQ 3.13, Redis 7, PostgreSQL 16 e uma pilha de observabilidade com Prometheus 2.52, Loki 2.9, Tempo 2.5 e Grafana 10.4.

A camada agentiva é implementada pelo módulo *agent-orchestrator*, desenvolvido em Spring Boot com LangChain4j 0.32. O modelo foi selecionado por ser de código aberto, executável em infraestrutura local e compatível com cenários em que a privacidade operacional limita o uso de LLMs proprietários hospedados em nuvem. A Tabela 2 resume os principais parâmetros do experimento.

6.2 Materialização da GTAA-LM na Prova de Conceito

A prova de conceito (Proof of Concept, PoC) materializa as camadas da GTAA-LM por meio de componentes executáveis e serviços

Tabela 2: Parâmetros de configuração do experimento.

Parâmetro	Valor
Modelo	*llama3:8b* via Ollama
Temperatura / top-p	0,1 / 0,9
Janela de contexto	8.192 tokens
Tokens máximos	2.048 por chamada
Iterações ReAct	máximo de 12
Mínimo de ferramentas	3 antes de *Final Answer*
Rodadas executadas	30 rodadas completas, com 6 cenários cada

de infraestrutura. A Figura 3 apresenta essa correspondência, distinguindo a arquitetura de referência, descrita na Seção 5, de sua instanciação utilizada na avaliação experimental.

Na camada de microsserviços, a PoC utiliza os serviços *api-gateway*, *order-service*, *payment-service*, *inventory-service* e *notification-service*, além de PostgreSQL, Redis e RabbitMQ como componentes de infraestrutura associados ao ambiente observado. A camada de observabilidade é materializada por Prometheus, Loki, Tempo e Grafana, responsáveis por disponibilizar métricas, logs, rastros e informações operacionais utilizadas durante o diagnóstico.

A camada de ferramentas e conectores é implementada pelas operações disponibilizadas ao agente para consulta ao ambiente, incluindo verificação de saúde dos serviços, recuperação de logs, latência e códigos HTTP, inspeção de filas, consulta à topologia e dependências entre serviços e recuperação do histórico de incidentes. Essas ferramentas constituem a única interface utilizada pelo componente agentivo para acessar evidências operacionais, evitando acesso direto do LLM à infraestrutura observada.

A camada agentiva é materializada pelo módulo *agent-orchestrator*, que contém o *ReActDiagnosticAgent* e utiliza o modelo *llama3:8b* executado externamente por meio do Ollama. Embora a arquitetura de referência defina responsabilidades agentivas especializadas, a instanciação avaliada consolida coordenação, diagnóstico, análise de dependências e consulta à documentação em um único agente ReAct. Essa decisão preserva a separação lógica das responsabilidades no *prompt*, nas ferramentas e no fluxo de execução, sem exigir múltiplos agentes autônomos na PoC.

A camada de governança e controle é materializada pelo *GovernanceService* e pelos mecanismos de auditoria associados às chamadas de ferramentas e às respostas produzidas. Esses componentes calculam indicadores de rastreabilidade, registram as interações e verificam a necessidade de aprovação humana para recomendações que podem produzir impacto operacional. Dessa forma, a decisão sobre supervisão não é delegada exclusivamente ao modelo de linguagem.

Por fim, a interação com o componente agentivo na PoC ocorre por meio do endpoint *POST /api/diagnose* e das respostas estruturadas produzidas pelo fluxo de diagnóstico. A instanciação experimental não implementa um painel Web completo da camada humano-agente; essa camada é representada pelo mecanismo de submissão da solicitação, pela resposta estruturada, pelos registros de auditoria e pela sinalização da necessidade de aprovação humana.

6.3 Agente ReAct e Ferramentas

O *ReActDiagnosticAgent* implementa manualmente o padrão ReAct (*Reasoning + Acting*). Em cada iteração, o modelo produz uma ação; o código Java executa a ferramenta real e injeta o resultado como *Observation*; o ciclo prossegue até a resposta JSON final. Essa implementação reduz a dependência de *tool-calling* nativo e mantém explícitos o acionamento das ferramentas e o registro das evidências.

A centralização das responsabilidades em um único agente foi uma escolha de engenharia associada ao uso de um modelo local de 8 bilhões de parâmetros e à necessidade de previsibilidade operacional. A separação arquitetural foi preservada por ferramentas, *prompts* estruturados e governança externa; os resultados não permitem concluir superioridade sobre coordenação multiagente. As ferramentas implementadas são *checkAllServicesHealth*, *queryLogs*, *queryLatency*, *queryHttpStatus*, *getServiceDependencies*, *getServiceTopology*, *inspectQueues* e *getIncidentHistory*, com parâmetros, saídas e duração registrados em auditoria.

6.4 Cenários de Falha e Procedimento

Os seis cenários da Tabela 3 exercitam classes distintas de problemas: indisponibilidade síncrona (S1), alta latência (S2), acúmulo de fila (S3), configuração incorreta (S4), autenticação (S5) e indisponibilidade de persistência (S6). S1 e S5 afetam deliberadamente o *payment-service*, permitindo verificar se o diagnóstico distingue HTTP 503 de HTTP 401 em vez de apenas localizar o componente relacionado ao sintoma.

Os demais cenários ampliam a diversidade das evidências necessárias ao diagnóstico: S2 mantém o serviço disponível, mas degrada seu desempenho; S3 exige inspeção do estado da mensageria; S4 introduz uma falha de configuração entre serviços; e S6 afeta a persistência compartilhada. Assim, a avaliação não se limita a indisponibilidade binária e exige combinações distintas de *health checks*, códigos HTTP, métricas, logs, filas e dependências. S6 também permite observar o comportamento da própria infraestrutura de auditoria quando um componente de suporte se torna indisponível.

Cada execução estabeleceu uma linha de base, injetou a falha, gerou tráfego sintético e então acionou *POST /api/diagnose*. Durante o diagnóstico, as interações com ferramentas e evidências eram registradas; ao final, a saída era comparada ao diagnóstico esperado. Os cenários foram executados isoladamente e o estado restaurado entre execuções. Foram realizadas 30 rodadas completas, totalizando 180 sessões independentes, permitindo observar estabilidade e variabilidade das respostas sob condições repetidas.

A restauração do ambiente entre cenários reduz interferências entre falhas consecutivas e mantém conhecida a causa de referência. Durante o ciclo ReAct, cada *Observation* retornada por uma ferramenta é incorporada ao contexto da iteração seguinte, de modo que a resposta final possa ser relacionada à sequência efetiva de consultas. Esse procedimento também permite distinguir uma limitação do modelo de uma limitação das evidências disponibilizadas pelas integrações experimentais.

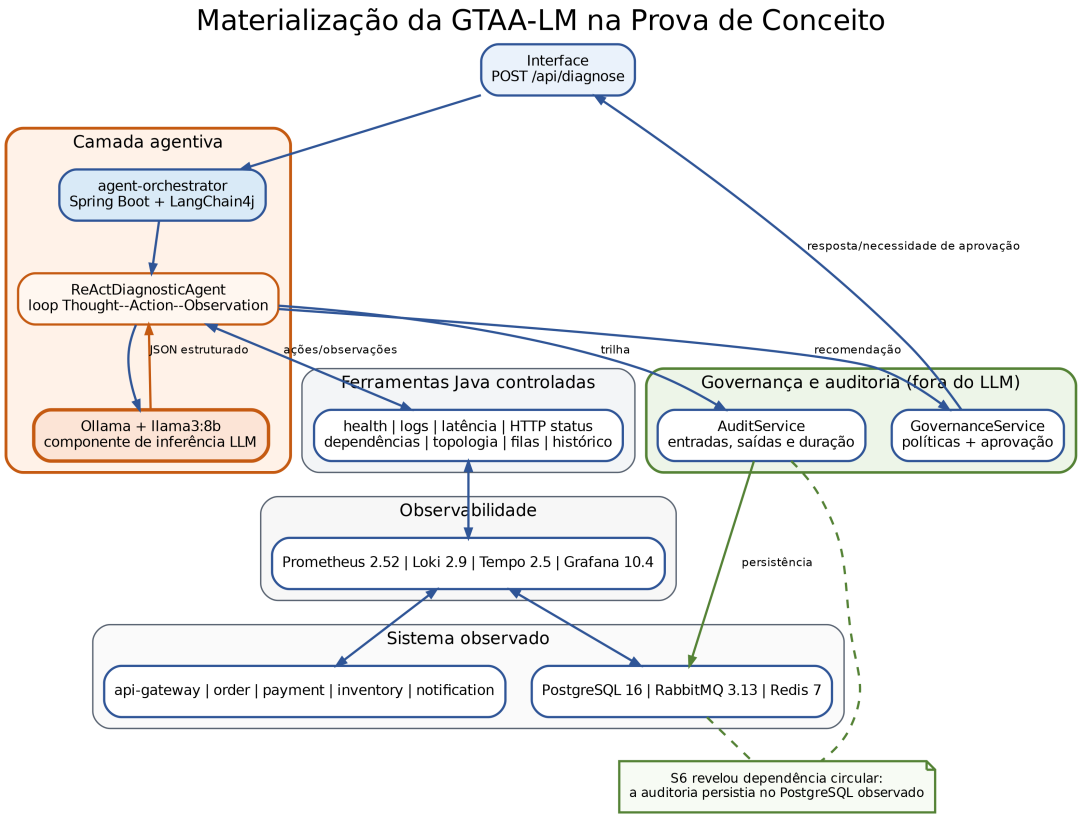

Figura 3: Materialização da GTAA-LM na prova de conceito. A figura relaciona as camadas da arquitetura de referência aos componentes utilizados na avaliação experimental, incluindo microsserviços, infraestrutura de observabilidade, ferramentas controladas, agente ReAct com LLM e mecanismos externos de governança e auditoria.

Tabela 3: Cenários de falha executados na prova de conceito.

ID	Cenário	Falha injetada	Evidência esperada	Diagnóstico esperado
S1	Pagamento indisponível	*payment-service* retorna HTTP 503.	Health check *down*; HTTP 503; falha no checkout.	Dependência crítica indisponível.
S2	Alta latência no estoque	Atraso artificial de 3.000 ms no *inventory-service*.	P99 elevado; traces lentos; serviço *up*.	Gargalo de desempenho.
S3	Fila acumulada	*notification-service* interrompido.	Fila acumulada e 0 consumidores.	Gargalo assíncrono.
S4	Erro de configuração	URL inválida do *inventory-service* no *order-service*.	Conexão recusada ou host inválido.	Falha de configuração.
S5	Falha de autenticação	*payment-service* retorna HTTP 401.	HTTP 401; token inter-serviço inválido.	Falha de autenticação/autorização.
S6	Banco indisponível	PostgreSQL interrompido.	Erros JDBC/SQL e health checks 503.	Falha de persistência.

6.5 Critérios de Avaliação e Tratamento de Alucinações

Uma resposta foi considerada correta quando identificou o serviço afetado e a causa provável com vocabulário compatível com o diagnóstico esperado. A verificação da causa utilizou expressões regulares com múltiplos sinônimos para acomodar variação lexical.

O *score* de rastreabilidade, calculado pelo *GovernanceService*, combina chamadas de ferramenta, evidências citadas e presença dos campos estruturais obrigatórios, este indicador mede a aderência ao protocolo de rastreabilidade da arquitetura e deve ser interpretado separadamente da correção do diagnóstico.

A PoC não implementa um detector específico de alucinações, e a mitigação ocorre arquiteturalmente: fatos sobre o estado do

sistema são recuperados por ferramentas e incorporados como *Observations*; a saída final é estruturada e submetida a verificações determinísticas; e recomendações com impacto operacional são sinalizadas para aprovação humana. Esses mecanismos separam evidências do ambiente das hipóteses formuladas pelo modelo, mas não eliminam interpretações causais incorretas.

Por isso, respostas incorretas foram contabilizadas como falhas de diagnóstico, sem classificação posterior entre alucinação, interpretação incompleta ou ambiguidade das evidências. Os resultados permitem avaliar correção operacional, mas não estimar uma taxa específica de alucinação.

6.6 Resultados Agregados

Os resultados agregados das 30 rodadas são apresentados na Tabela 4. O agente identificou corretamente o serviço afetado em 180/180 execuções (100%). Esse resultado não representa uma taxa de 100% de diagnóstico correto: quando também se exige a identificação adequada da causa provável, o diagnóstico completo foi correto em 112/180 execuções (62,2%). Portanto, os resultados distinguem explicitamente a localização do componente afetado da capacidade de inferir corretamente a causa da falha. A rastreabilidade média global foi 0,945 ± 0,120, com média de 0,995 ± 0,027 quando o cenário S6 é excluído. O tempo médio de resposta foi 15,23 ± 10,03 s. A camada de governança sinalizou corretamente a necessidade de aprovação humana em 180/180 execuções.

A evolução dos indicadores por rodada foi mantida nos artefatos complementares. Em termos agregados, a identificação de serviço permaneceu estável, enquanto o diagnóstico completo apresentou variação entre rodadas, reforçando a natureza exploratória da avaliação.

Os cenários S2, S3 e S4 apresentaram os melhores resultados porque as evidências observáveis eram semanticamente distintas: latência elevada, fila sem consumidores e erro de configuração, respectivamente. Em contraste, S1 e S5 apresentaram menor taxa de diagnóstico completo. Embora o serviço afetado tenha sido corretamente identificado em todas as execuções, o modelo frequentemente produziu causas genéricas para falhas do *payment-service*, especialmente quando os logs consultados não retornavam evidências suficientemente estruturadas para diferenciar indisponibilidade (503) de autenticação (401). Esse resultado demonstra que a precisão do diagnóstico depende não apenas do LLM, mas da qualidade, granularidade e acessibilidade das evidências observáveis.

Esses casos também ilustram por que a identificação correta do serviço não deve ser interpretada como garantia de confiabilidade causal da resposta. O modelo pôde localizar corretamente o componente relacionado ao incidente e, ainda assim, formular uma causa genérica ou incompatível com a falha injetada. Como não foi realizada uma anotação específica de alucinações, esses casos são tratados neste estudo como erros de diagnóstico causal.

6.7 Análise dos Resultados e Relação com as RQs

Em relação à RQ1, os resultados mostram que a GTAA-LM pode integrar um componente agentivo baseado em LLM a um ambiente distribuído por meio de ferramentas controladas e fontes externas de evidência. A separação entre inferência, acesso ao ambiente e governança mostrou-se tecnicamente viável na instanciação avaliada. A principal condição observada é que a qualidade do diagnóstico permanece diretamente relacionada à qualidade e à granularidade das evidências disponibilizadas pelas ferramentas.

Quanto à RQ2, a prova de conceito mostrou que rastreabilidade, controle de ferramentas e supervisão humana podem ser implementados como mecanismos externos ao LLM. As interações foram associadas às evidências e ferramentas utilizadas, enquanto recomendações com potencial impacto operacional foram submetidas às políticas de governança. O cenário S6, entretanto, revelou que a efetividade desses mecanismos também depende da resiliência da própria infraestrutura utilizada para auditoria.

Por fim, em relação à RQ3, a avaliação evidencia que a arquitetura é capaz de apoiar a localização do componente associado ao incidente, mas a inferência da causa depende de evidências suficientemente discriminantes. Os cenários em que sintomas semelhantes estavam associados a causas distintas mostraram-se particularmente desafiadores, indicando que a utilidade do componente baseado em LLM não deve ser avaliada apenas pela identificação do serviço afetado, mas também pela correção causal e pelo suporte das recomendações nas evidências coletadas.

6.8 S6 como Resultado Arquitetural

O cenário S6 constitui o resultado arquitetural mais relevante da avaliação. O agente identificou corretamente o serviço afetado em todas as execuções, mas a rastreabilidade média caiu para 0,697 ± 0,093 e as chamadas de ferramenta deixaram de ser registradas na maior parte das rodadas. A causa raiz é uma dependência circular: o *AuditService* persiste registros no PostgreSQL, que é exatamente o componente derrubado no cenário S6. Assim, o mecanismo de rastreabilidade torna-se indisponível durante uma falha de infraestrutura em que a auditoria seria especialmente necessária.

Esse comportamento constitui uma evidência arquitetural obtida durante a avaliação: mecanismos responsáveis pela rastreabilidade não devem compartilhar um ponto de falha com a infraestrutura que monitoram. A GTAA-LM deve, portanto, incorporar como requisito o isolamento do mecanismo de auditoria. Estratégias possíveis incluem fila assíncrona durável para eventos de auditoria, persistência local temporária em arquivo *append-only*, *backend* independente do banco transacional observado ou armazenamento imutável sincronizado após a recuperação do serviço.

7 Discussão

A avaliação da GTAA-LM evidencia que a principal contribuição da arquitetura não está no desempenho isolado do modelo de linguagem, mas na definição das fronteiras entre inferência probabilística, evidências operacionais, ferramentas controladas e mecanismos determinísticos de governança. Os experimentos mostraram que a localização do componente associado ao incidente pode ser consistente mesmo quando a identificação da causa permanece limitada, especialmente quando as evidências disponíveis não distinguem adequadamente diferentes classes de falha.

Sob a perspectiva de confiabilidade do LLM, a vinculação a ferramentas reduz a necessidade de inferir o estado do sistema exclusivamente a partir do conhecimento paramétrico do modelo, mas

Tabela 4: Resultados agregados por cenário em 30 rodadas independentes.

ID	Serviço correto	Diagnóstico completo	Rastreabilidade média	Tempo médio	Aprovação humana	n
S1	100%	3,3% (1/30)	0,995 ± 0,027	10,75 ± 1,24 s	30/30	30
S2	100%	96,7% (29/30)	1,000 ± 0,000	11,43 ± 2,51 s	30/30	30
S3	100%	100% (30/30)	0,995 ± 0,027	12,64 ± 3,04 s	30/30	30
S4	100%	100% (30/30)	1,000 ± 0,000	10,40 ± 0,74 s	30/30	30
S5	100%	13,3% (4/30)	0,985 ± 0,046	11,60 ± 1,79 s	30/30	30
S6	100%	60,0% (18/30)	0,697 ± 0,093	34,55 ± 11,54 s	30/30	30
Total	100%	62,2% (112/180)	0,945 ± 0,120	15,23 ± 10,03 s	180/180	180

não elimina erros na interpretação causal das evidências. Por essa razão, a GTAA-LM trata informações recuperadas do ambiente e inferências produzidas pelo modelo como elementos distintos: as primeiras constituem evidências verificáveis, enquanto as segundas permanecem hipóteses ou recomendações sujeitas a validação, auditoria e, quando necessário, supervisão humana.

Outro resultado relevante decorre da relação entre arquitetura de referência e sua instanciação. Embora a GTAA-LM defina responsabilidades agentivas especializadas, a prova de conceito consolidou essas responsabilidades em um único agente ReAct. Essa decisão demonstra que a separação arquitetural de responsabilidades não implica necessariamente a implantação de múltiplos agentes autônomos. Os resultados, contudo, não permitem comparar a eficácia dessa opção com arquiteturas multiagentes, questão que permanece aberta para avaliações futuras.

A governança externa ao LLM também mostrou-se importante para limitar a autonomia do componente probabilístico. Políticas de autorização, registro de evidências, auditoria e aprovação humana podem ser aplicadas independentemente do conteúdo produzido pelo modelo. Essa separação favorece uma interpretação da GTAA-LM como arquitetura de apoio à tomada de decisão, e não como mecanismo de operação autônoma de infraestrutura.

Finalmente, o comportamento observado no cenário S6 demonstra que governança e observabilidade também precisam ser projetadas para condições de falha. Uma trilha de auditoria é particularmente necessária durante incidentes graves e, portanto, não deve depender exclusivamente dos mesmos componentes cuja indisponibilidade pretende registrar. Esse resultado introduz um requisito adicional de resiliência para futuras instanciações da arquitetura.

8 Limitações e Trabalhos Futuros

Os resultados apresentados neste artigo devem ser interpretados à luz do escopo exploratório da prova de conceito. O ambiente experimental foi conduzido de forma controlada, com cinco microsserviços e seis cenários de falha previamente definidos. Embora esse ambiente represente padrões comuns de sistemas distribuídos, trabalhos futuros devem ampliar a avaliação para arquiteturas com maior diversidade de serviços, políticas de implantação, volumes de tráfego, mecanismos de autenticação, filas, bancos de dados e níveis de maturidade em observabilidade.

Outra direção relevante consiste em comparar diferentes modelos LLM, tamanhos e provedores. Esta avaliação utilizou o modelo local *llama3:8b*, escolha coerente com o objetivo de investigar uma solução executável em infraestrutura local e com maior controle de privacidade. No entanto, modelos maiores, especializados ou com suporte nativo a *tool-calling* podem apresentar comportamento distinto em termos de precisão causal, estabilidade, custo computacional e capacidade de uso de ferramentas. Avaliações futuras devem, portanto, investigar a sensibilidade da GTAA-LM a diferentes modelos e configurações.

Também é necessário aprimorar os critérios de avaliação do diagnóstico. Nesta prova de conceito, a verificação da causa provável foi baseada em expressões regulares e palavras-chave por cenário, com múltiplos sinônimos para acomodar variação lexical. Esse procedimento favorece reprodutibilidade, mas pode subestimar diagnósticos semanticamente corretos que utilizem vocabulário não previsto ou superestimar respostas que contenham termos esperados sem explicação causal suficiente. Como trabalho futuro, pretende-se combinar verificação automática com julgamento independente por especialistas, considerando utilidade técnica, completude, explicabilidade e adequação das recomendações.

Outra limitação é que o experimento não realizou classificação independente das respostas quanto à ocorrência de alucinações. Embora a arquitetura restrinja o acesso ao ambiente por ferramentas, preserve as evidências recuperadas e aplique validações determinísticas sobre a saída, o LLM ainda pode estabelecer relações causais não sustentadas ou interpretar incorretamente informações válidas. Neste estudo, tais situações são refletidas indiretamente pela métrica de diagnóstico completo, mas não permitem estimar uma taxa específica de alucinação. Trabalhos futuros devem incorporar anotação independente das respostas e critérios explícitos para distinguir afirmações suportadas, não suportadas e contraditórias em relação às evidências coletadas.

A evolução da infraestrutura de observabilidade também constitui uma direção importante. Os cenários S1 e S5 indicaram que, quando logs e métricas não retornam evidências específicas e estruturadas, o agente tende a identificar corretamente o serviço afetado, mas pode produzir causas genéricas. Assim, versões futuras da prova de conceito devem incorporar logs mais estruturados, melhor padronização semântica de eventos, correlação com traces distribuídos, enriquecimento de métricas e mecanismos de recuperação contextual a partir de documentação arquitetural.

A resiliência da auditoria constitui ainda uma direção arquitetural prioritária. A partir da limitação evidenciada em S6, futuras instanciações devem desacoplar a persistência da trilha de auditoria da infraestrutura transacional observada, investigando mecanismos

duráveis e independentes para preservação dos registros durante falhas.

9 Considerações Finais

Este artigo apresentou a GTAA-LM, uma arquitetura de referência para a integração controlada, rastreável e supervisionada de agentes baseados em LLM em sistemas distribuídos de microsserviços. A proposta organiza explicitamente as responsabilidades de observabilidade, acesso a ferramentas, inferência baseada em LLM, governança, auditoria e interação humana, evitando que o modelo de linguagem possua acesso irrestrito ao ambiente operacional.

A instanciação experimental demonstrou a viabilidade da arquitetura como mecanismo de apoio ao diagnóstico, ao mesmo tempo em que revelou limitações na inferência causal quando as evidências disponíveis são insuficientemente discriminantes. Os experimentos também evidenciaram que mecanismos de rastreabilidade precisam ser projetados de forma resiliente e independente dos componentes observados.

Em conjunto, os resultados sustentam o uso da GTAA-LM como uma arquitetura de apoio à análise operacional, e não como evidência de autonomia plena de agentes sobre sistemas distribuídos. A contribuição central está em estabelecer fronteiras arquiteturais que permitam combinar capacidades de interpretação dos LLMs com evidências verificáveis, ferramentas controladas, políticas de governança e supervisão humana.

Disponibilidade de Artefatos

Os artefatos utilizados na avaliação da GTAA-LM, incluindo código-fonte, configurações, scripts de execução e injeção de falhas, cenários, resultados experimentais, respostas do agente e trilhas de auditoria, estão preservados em uma versão específica e persistente no Zenodo [9]. O artefato está disponível em: https://doi.org/10.5281/zenodo.22146387.

Declaração sobre o Uso de IA Generativa

Ferramentas de IA generativa foram utilizadas como apoio à revisão linguística, reorganização textual e elaboração preliminar de elementos gráficos. A definição do problema, a arquitetura, a implementação, os experimentos, a análise dos resultados e a validação científica foram realizadas e revisadas pelos autores, que assumem integral responsabilidade pelo conteúdo do artigo.

Agradecimentos

Este trabalho recebeu apoio da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), projeto APQ-00996-24; da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processos 2025/09390-3 e 2021/06968-3; da Universidade Federal de Lavras (UFLA), por meio do Auxílio n.º 014/2026 – Inovação UFLA, Processo n.º 23090.000597/2026-73; do Ministério das Comunicações, por meio do TED 136/2022; da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES), por meio de financiamento institucional – Código 001; do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo apoio financeiro parcial; e da Financiadora de Estudos e Projetos (Finep).

Referências

[1] Qian Cheng, Doyen Sahoo, Amrita Saha, Wenzhuo Yang, Chenghao Liu, Gerald Woo, Manpreet Singh, Silvio Savarese, and Steven C. H. Hoi. 2023. AI for IT Operations (AIOps) on Cloud Platforms: Reviews, Opportunities and Challenges. arXiv:2304.04661 [cs.SE] https://arxiv.org/abs/2304.04661

[2] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2019. Architecting with Microservices: A Systematic Mapping Study. *Journal of Systems and Software* 150 (2019), 77–97. doi:10.1016/j.jss.2019.01.001

[3] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2017. Microservices: Yesterday, Today, and Tomorrow. In *Present and Ulterior Software Engineering*. Springer, Cham, 195–216. doi:10.1007/978-3-319-67425-4_12

[4] Cuiyun Gao, Xing Hu, Shan Gao, Xin Xia, and Zhi Jin. 2025. The Current Challenges of Software Engineering in the Era of Large Language Models. *ACM Transactions on Software Engineering and Methodology* 34, 5, Article 127 (2025), 30 pages. doi:10.1145/3712005

[5] Junda He, Christoph Treude, and David Lo. 2025. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Transactions on Software Engineering and Methodology* 34, 5, Article 124 (2025), 30 pages. doi:10.1145/3712003

[6] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* (2024). doi:10.1145/3695988

[7] Dae-Kyoo Kim and Hua Ming. 2025. Assessing Output Reliability and Similarity of Large Language Models in Software Development: A Comparative Case Study Approach. *Information and Software Technology* 185 (2025), 107787. doi:10.1016/j.infsof.2025.107787

[8] Miles Q. Li and Benjamin C. M. Fung. 2025. Security Concerns for Large Language Models: A Survey. *Journal of Information Security and Applications* 95 (2025), 104284. doi:10.1016/j.jisa.2025.104284

[9] Júlio C. S. Lima, Luiz Henrique Andrade Correia, André de Lima Salgado, and Maurício Ronny de Almeida Souza. 2026. *GTAA-LM: Artifact and Experimental Data.* doi:10.5281/zenodo.22146387

[10] Junwei Liu, Kaixin Wang, Yixuan Chen, Xin Peng, Zhenpeng Chen, Lingming Zhang, and Yiling Lou. 2026. Large Language Model-Based Agents for Software Engineering: A Survey. *ACM Transactions on Software Engineering and Methodology* (2026). doi:10.1145/3796507

[11] Wenli Yang, Lilian Some, Michael Bain, and Byeong Kang. 2025. A Comprehensive Survey on Integrating Large Language Models with Knowledge-Based Methods. *Knowledge-Based Systems* (2025), 113503. doi:10.1016/j.knosys.2025.113503

[12] Qingyang Yu, Nengwen Zhao, Mingjie Li, Zeyan Li, Junjie Wang, Xiaohui Li, Qingsong He, Xiaofei Sun, Yongqiang Liu, Dongmei Zhang, and Dan Pei. 2024. A Survey on Intelligent Management of Alerts and Incidents in IT Services. *Journal of Network and Computer Applications* 224 (2024), 103842. doi:10.1016/j.jnca.2024.103842

[13] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Aiwei Liu, Yong Yang, Zhonghai Wu, Xuming Hu, Philip S. Yu, and Ying Li. 2025. A Survey of AIOps in the Era of Large Language Models. *Comput. Surveys* (2025). doi:10.1145/3746635

[14] Quanjun Zhang, Chunrong Fang, Yuxiang Xie, Yong Zhang, Yun Yang, Weifeng Sun, Shanshan Yu, and Zhenyu Chen. 2026. A Survey on Large Language Models for Software Engineering. *Science China Information Sciences* (2026). doi:10.1007/s11432-025-4670-0