
	<p style="text-align: center;">UNIVERSIDADE ESTADUAL DO CEARÁ</p> <p style="text-align: center;">Programa de Pós-Graduação em Ciência da Computação</p> <p style="text-align: center;">Tópicos em IA (NLP)</p> <p style="text-align: center;">Profº Leonardo Sampaio Rocha</p> <p style="text-align: center;">Aluno: Anderson Martins Gomes</p>	
---	---	---

PROJETO FINAL - Proposta - 09/06/2025

Agentes Baseados em LLM como Entidades Autônomas de Negócio: Uma Nova Arquitetura para Construção Adaptativa de Sistemas de Informação

1 RESUMO

Este trabalho propõe uma arquitetura multiagente baseada em modelos de linguagem pré-treinados (LLMs), com foco na construção e evolução adaptativa de sistemas de informação. Diferentemente das abordagens tradicionais que simulam papéis clássicos da engenharia de software — como desenvolvedores, arquitetos ou engenheiros de testes — por meio de agentes LLM, introduzimos uma nova camada conceitual denominada *Business Autonomous Entities* (BAEs). Cada BAE representa uma entidade de domínio (como “Aluno”, “Disciplina” ou “Professor”) e é responsável por sua modelagem semântica, persistência de dados, geração de interfaces e coordenação com agentes auxiliares.

A arquitetura é composta por quatro elementos principais: *Human Business Experts* (HBEs), que interagem com o sistema por meio de linguagem natural; *Human Software Engineering Experts* (HSWEs), que oferecem suporte técnico quando necessário; *Business Autonomous Entities* (BAEs), responsáveis por entidades de negócio específicas; e *Software Engineering Autonomous Agents* (SWEAs), que realizam tarefas de engenharia de software sob demanda. A interação dinâmica entre esses elementos permite a construção incremental e adaptativa do sistema, rompendo com a separação rígida entre as fases de projeto e execução.

Como exemplo prático, considere a criação e evolução da entidade “Aluno” em um sistema acadêmico. Inicialmente, um Human Business Expert (HBE) — como um coordenador de curso — interage com o sistema por meio de linguagem natural, solicitando: “Crie uma entidade de aluno com nome completo, matrícula e curso.” A Business Autonomous Entity (BAE) correspondente à entidade “Aluno” interpreta esse comando, define uma estrutura inicial e aciona um Software Engineering Autonomous Agent (SWEA) programador para gerar o código do modelo de dados, a API com operações CRUD e o script de persistência em banco de dados. Em seguida, outro SWEA é responsável por construir automaticamente uma interface para inserção e visualização de alunos.

Posteriormente, o HBE percebe a necessidade de capturar informações adicionais e solicita: “Adicione a data de ingresso e o coeficiente de rendimento ao aluno.” A mesma BAE reconhece que se trata de uma evolução do modelo e atualiza sua representação semântica da entidade. Com base nisso, ela coordena novamente os SWEAs: o programador gera a nova versão do modelo e ajusta os endpoints da API, enquanto o SWEA de frontend atualiza dinamicamente a interface do usuário para refletir os novos campos. Se qualquer mudança impactar aspectos técnicos mais delicados, como integridade de dados ou compatibilidade com versões anteriores, o Human Software Engineering Expert (HSWE) pode ser envolvido para tomar decisões críticas. Esse ciclo demonstra como os quatro elementos da arquitetura operam de forma colaborativa e contínua para transformar, adaptar e evoluir um sistema de forma incremental, mantendo a coerência entre o vocabulário de negócio e os artefatos técnicos gerados.

Além de sua atuação contextualizada em tempo de execução, os BAEs oferecem um enorme potencial de reutilização entre diferentes sistemas, organizações ou domínios com estruturas semelhantes. Como encapsulam o conhecimento operacional e estrutural de uma entidade de negócio, esses agentes podem ser armazenados e distribuídos em bibliotecas reutilizáveis — públicas ou privadas — que evoluem de forma colaborativa conforme a maturidade do domínio modelado. Por exemplo, um BAE da entidade “Aluno” criado em uma universidade pode ser facilmente reaproveitado por outras instituições de ensino, exigindo apenas configurações locais para adequação a suas regras específicas. Essa abordagem

favorece a padronização de entidades de domínio e promove um ecossistema de agentes especializados que podem ser integrados modularmente.

Ainda mais, a ideia de reutilização de BAEs guarda forte similaridade com os princípios da orientação a objetos, em que classes e instâncias encapsulam atributos e comportamentos. Nesse sentido, BAEs podem ser vistos como "objetos inteligentes" e autônomos, com responsabilidades bem definidas e capacidade de adaptação, herdando e estendendo funcionalidades de entidades genéricas para contextos mais específicos — como, por exemplo, um BAE “Aluno de Pós-Graduação” que herda e expande um BAE genérico “Aluno”. Essa analogia reforça o alinhamento da arquitetura proposta com práticas consolidadas de engenharia de software, ao mesmo tempo em que amplia sua aplicabilidade prática e escalabilidade para contextos corporativos, acadêmicos e governamentais.

2 QUESTÕES/HIPÓTESES DE PESQUISA

- **RQ1:** É possível estruturar agentes baseados em LLM que representem entidades de negócio de forma reutilizável e configurável, independentes do contexto de aplicação?
- **RQ2:** Qual o nível de autonomia que esses agentes podem alcançar na geração e adaptação de sistemas sem intervenção humana especializada?
- **RQ3:** Como a complexidade e o custo de desenvolvimento de sistemas utilizando BAEs se comparam às abordagens atuais baseadas em agentes LLM tradicionais?

3 TRABALHOS RELACIONADOS

A arquitetura proposta se destaca em relação aos trabalhos existentes tanto em sistemas multiagente LLM quanto em práticas clássicas de modelagem de domínio. As abordagens de He et al. (2024) modelam papéis típicos de engenharia de software, como desenvolvedores e testadores, operando no design time, mas não consideram a evolução *runtime* do domínio. Em contraste, nossa proposta integra uma camada inovadora de *Business Autonomous Entities* (BAEs) que representam entidades de negócio — como Aluno ou Disciplina — dentro do próprio sistema, permitindo modelagem viva, persistência, interface e adaptação contínua.

Tal ideia se alinha aos frameworks adaptativos em agente como o HYDRA (Mohan, Shiwali, et al., 2024), que prevê a detecção dinâmica de mudanças ambientais, e respeita princípios de *Domain-Driven Design* (DDD), onde contextos limitados (*bounded contexts*) se tornam módulos com representação cognitiva — conforme discutido por Ricci et al. (2024). Além disso, nossa proposta aproveita as capacidades de orquestração com LangChain/LangGraph (Mavroudis, 2024), adotando uma arquitetura modular e rastreável de agentes, embora vá além ao incorporar runtime evolutivo e reuso semântico através de BAEs.

Por fim, se diferencia de trabalhos emergentes como ChatDev (Qian et al., 2023), AgentVerse (Chen, Weize, et al, 2023) ao descentralizar o controle em entidades de domínio autônomas, alinhando o sistema com cenários *low-code/no-code* inteligentes e agentes reutilizáveis — marcando um avanço em reuso de conhecimento, arquitetura adaptativa e autonomia contextualizada.

4 POSSÍVEIS APLICAÇÕES

A arquitetura proposta possui amplo potencial de aplicação em contextos nos quais sistemas de informação precisam ser desenvolvidos ou adaptados rapidamente com base em regras de negócio específicas. Um dos cenários mais promissores é o de plataformas de desenvolvimento automatizado de sistemas administrativos, como ERPs e sistemas corporativos, nos quais entidades como “Aluno”, “Cliente” ou “Produto” compartilham estruturas recorrentes, mas exigem parametrizações conforme o domínio. A proposta também viabiliza a criação de ferramentas para prototipagem rápida e geração de MVPs, permitindo que especialistas de negócio, mesmo sem formação técnica, descrevam suas necessidades em linguagem natural e obtenham sistemas funcionais.

Além disso, o modelo pode servir como base para plataformas *low-code* ou *no-code* baseadas em agentes inteligentes, promovendo a democratização do desenvolvimento de software. Na pesquisa e na educação, BAEs reutilizáveis podem acelerar a construção de simuladores e sistemas experimentais personalizados.

Por fim, estima-se que a arquitetura proposta contribuirá para a redução de custos e complexidade no desenvolvimento de sistemas em ambientes com alta

rotatividade de requisitos, estabelecendo um novo paradigma de geração contínua e adaptativa de aplicações.

5 PROVA DE CONCEITO

Para validar a viabilidade da arquitetura proposta, será desenvolvida uma prova de conceito centrada na implementação de uma Business Autonomous Entity (BAE) responsável pela entidade "Aluno" em um sistema acadêmico simulado. O objetivo principal é demonstrar que os BAEs podem operar de forma autônoma na geração e evolução de sistemas, mantendo coerência semântica entre o domínio de negócio e os artefatos técnicos produzidos.

A prova de conceito será estruturada em três cenários progressivos de complexidade. No **Cenário 1 - Geração Inicial**, um Human Business Expert (HBE) fornecerá uma solicitação em linguagem natural como "Crie um sistema para gerenciar alunos com nome, matrícula e curso". A BAE "Aluno" deverá interpretar essa demanda, definir a estrutura semântica da entidade e orquestrar Software Engineering Autonomous Agents (SWEAs) para gerar automaticamente: (i) um modelo de dados Pydantic, (ii) uma API REST funcional com operações CRUD, (iii) um banco de dados SQLite com esquema apropriado, e (iv) uma interface web básica para manipulação dos dados. O sistema resultante deverá estar operacional e acessível via navegador.

No **Cenário 2 - Evolução Runtime**, será testada a capacidade adaptativa da arquitetura. O HBE solicitará modificações como "Adicione os campos data de nascimento e coeficiente de rendimento ao aluno". A BAE deverá reconhecer que se trata de uma evolução do modelo existente, atualizar sua representação interna e coordenar os SWEAs para: (i) migrar o esquema do banco de dados preservando dados existentes, (ii) regenerar a API com os novos campos, (iii) atualizar a interface web dinamicamente, e (iv) manter a integridade referencial. Esse cenário validará a hipótese de evolução contínua em tempo de execução.

O **Cenário 3 - Reutilização e Configuração** demonstrará o potencial de reuso dos BAEs. A mesma BAE "Aluno" será instanciada em um contexto diferente (por exemplo, um sistema de cursos livres), exigindo apenas configurações específicas como "O aluno deve ter campo de modalidade (presencial/online) e não

precisa de matrícula formal". Este cenário evidenciará a capacidade de adaptação contextual sem necessidade de recodificação.

A implementação será realizada integralmente em Python, utilizando OpenAI GPT-4o-mini para o processamento de linguagem natural dos agentes, FastAPI para geração de APIs, Streamlit para interfaces web, SQLAlchemy para persistência e LangGraph para orquestração de agentes. A comunicação entre agentes seguirá um protocolo de mensagens JSON estruturadas, permitindo rastreabilidade completa das interações.

As métricas de avaliação incluirão: (i) **tempo de resposta** para geração inicial do sistema (objetivo: < 3 minutos), (ii) **acurácia semântica** na interpretação dos comandos de negócio (avaliação qualitativa), (iii) **taxa de sucesso** na evolução incremental (objetivo: 100% para modificações simples), (iv) **qualidade do código gerado** (análise de sintaxe e boas práticas), e (v) **grau de reutilização** entre contextos diferentes (medido pela proporção de configuração vs. recodificação necessária).

A prova de conceito será documentada com capturas de tela das interfaces geradas, logs das interações entre agentes, exemplos de código produzido automaticamente e análise qualitativa das limitações identificadas. Os resultados servirão como base empírica para validar as questões de pesquisa propostas e fundamentar discussões sobre escalabilidade, robustez e aplicabilidade prática da arquitetura BAE em cenários reais de desenvolvimento de software.

6 REFERÊNCIAS

Chen, Weize, et al. "Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents." arXiv preprint arXiv:2308.10848 2.4 (2023): 6.

He, Junda, Christoph Treude, and David Lo. "LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead." ACM Transactions on Software Engineering and Methodology (2024).

Mavroudis, Vasilios. "LangChain." (2024).

Mohan, Shiwali, et al. "A domain-independent agent architecture for adaptive operation in evolving open worlds." *Artificial Intelligence* 334 (2024): 104161.

Qian, Chen, et al. "Chatdev: Communicative agents for software development." *arXiv preprint arXiv:2307.07924* (2023).

Ricci, Alessandro, et al. "Agents for DDD—Back and Forth." *International Workshop on Engineering Multi-Agent Systems*. Cham: Springer Nature Switzerland, 2024.