# FERRAMENTA AUTOMÁTICA PARA GERAÇÃO DE DIAGRAMA DE CLASSES A PARTIR DOS REQUISITOS: Uma abordagem com LLM e Model-Driven Architecture aplicada a um estudo de caso na indústria

Eduarda M. De Souza, Pedro L. M. Fortunato, Maria Sophia Pardinho Da Silva, Gabriel Gonçalves Costa, Flávio Henrique de Oliveira.

## RESUMO

Este artigo investiga o uso de *Large Language Models* (LLMs) como apoio à transformação de requisitos funcionais textuais em artefatos formais, visando a geração automática de diagramas de classes UML. O estudo propõe e avalia uma metodologia baseada em *Model-Driven Architecture* (MDA), na qual os requisitos são elicitados em um contexto industrial real e estruturados como cenários de caso de uso. Em seguida, esses cenários são normalizados por uma LLM segundo restrições sintáticas baseadas em BNF e submetidos a um ponto de controle de conformidade para assegurar consistência, completude e aderência ao padrão. Na etapa final, cenários validados alimentam um gerador que aplica regras de transformação para produzir o diagrama de classes no nível *Platform Independent Model* (PIM), seguido de análise qualitativa quanto à coerência estrutural, consistência terminológica e aderência ao domínio. A abordagem foi aplicada em um estudo de caso na indústria Ucelo, cujo processo atual depende de planilhas complexas e pouco rastreáveis, e os resultados indicam viabilidade prática para reduzir ambiguidades e apoiar a modelagem, produzindo modelos compatíveis com a evolução futura para etapas PSM e geração de código. Os resultados indicam que a combinação de LLMs com uma metodologia orientada por *Model-Driven Architecture* é viável para reduzir ambiguidades e aumentar a padronização de requisitos, permitindo transformar cenários de caso de uso em modelos UML mais consistentes e rastreáveis.

**Palavras-chave:** Engenharia de requisitos. Model Driven Architecture. LLM. UML. Transformação CIM–PIM.

# AUTOMATIC TOOL FOR GENERATING CLASS DIAGRAMS FROM REQUIREMENTS: An LLM and Model-Driven Architecture approach applied to an industrial case study

Eduarda M. De Souza, Pedro L. M. Fortunato, Maria Sophia Pardinho Da Silva, Gabriel Gonçalves Costa, Flávio Henrique de Oliveira.

## ABSTRACT

This paper investigates the use of Large Language Models (LLMs) to support the transformation of textual functional requirements into formal artifacts, aiming at the automatic generation of UML class diagrams. The study proposes and evaluates a Model-Driven Architecture (MDA)-based methodology in which requirements are elicited in a real industrial context and structured as use case scenarios. Next, these scenarios are normalized by an LLM according to BNF-based syntactic constraints and submitted to a conformance checkpoint to ensure consistency, completeness, and adherence to the defined standard. In the final stage, validated scenarios feed a generator that applies transformation rules to produce the class diagram at the Platform Independent Model (PIM) level, followed by a qualitative analysis regarding structural coherence, terminological consistency, and domain alignment. The approach was applied in a case study in the Ucelo industry, whose current process relies on complex spreadsheets with limited traceability, and the results indicate practical feasibility to reduce ambiguities and support modeling, producing models compatible with future evolution toward PSM stages and code generation. Overall, the results suggest that combining LLMs with a Model-Driven Architecture-oriented methodology is a viable way to reduce ambiguity and increase requirements standardization, enabling the transformation of use case scenarios into more consistent and traceable UML models.

**Key-words:** Requirements Engineering. Model-Driven Architecture. LLM. UML. CIM–PIM Transformation.

## 1. INTRODUÇÃO

O avanço da inteligência artificial tem aumentado significativamente as possibilidades de facilitar diversas etapas no desenvolvimento de software. Como a LLM (*Large Language Models*), são capazes de compreender, interpretar e gerar textos complexos. A capacidade desses modelos de processar linguagem natural abre portas para facilitar tarefas manuais, como a análise e interpretação de requisitos funcionais.

O problema que orienta este estudo consiste na dificuldade recorrente enfrentada por engenheiros de requisitos ao traduzir requisitos funcionais textuais em diagramas de classes consistentes e alinhados ao padrão UML. Essa atividade, essencial para o correto entendimento do sistema, exige capacidade de interpretação, experiência prévia em modelagem e tempo considerável de análise. Além disso, inconsistências ou falhas na modelagem inicial podem gerar retrabalho e impactar diretamente a qualidade da aplicação final. Nesse contexto, compreender como uma LLM pode apoiar esse processo de forma rápida e eficiente representa um avanço significativo para o meio acadêmico.

O objetivo deste artigo é analisar e demonstrar a aplicação de uma LLM no suporte à transformação de requisitos funcionais em um formato estruturado baseado em BNF (*Backus–Naur Form*), permitindo que esse conteúdo estruturado seja posteriormente convertido em diagramas de classes UML. O estudo apresenta o processo adotado para que o modelo identifique classes, atributos e relacionamentos implícitos nos textos de requisitos e os traduza para uma gramática formal; descreve os critérios utilizados para avaliar a consistência e a precisão da estrutura gerada; e discute o potencial dessa abordagem como uma ferramenta complementar no processo de desenvolvimento de software.

A relevância deste estudo se justifica pela crescente adoção de LLMs em tarefas repetitivas e de suporte à engenharia de software, bem como pelo potencial de reduzir o tempo e os erros associados à fase de modelagem. Além disso, a pesquisa contribui para ampliar o entendimento sobre os limites e possibilidades da

IA generativa, especialmente no contexto de integração entre linguagem natural e representações formais utilizadas na engenharia de software.

## 2. OBJETIVO GERAL

O objetivo deste artigo consiste em conceber, implementar e avaliar uma metodologia baseada em *Model Driven Architecture*, apoiada por modelos de inteligência artificial, para estruturar e refinar requisitos em cenários de caso de uso e, a partir desses artefatos, viabilizar a geração automática de diagramas de classes em nível PIM, analisando a qualidade e a aderência do modelo produzido ao domínio do problema.

## 3. OBJETIVO ESPECÍFICO

- Levantar e consolidar os requisitos do problema junto à organização demandante por meio de técnicas de elicitação;
- Estruturar os requisitos em cenários de caso de uso, contemplando fluxo principal, fluxos alternativos/exceção;
- Definir estruturas frasais e estratégias de engenharia de prompt para orientar o refinamento dos cenários por uma LLM;
- Refinar e padronizar os cenários com apoio da LLM, assegurando conformidade com regras de transformação no contexto de MDA;
- Submeter cenário as regras de transformação para gerar automaticamente o diagrama de classes em nível PIM a partir dos cenários validados;
- Avaliar qualitativamente o diagrama gerado quanto à coerência estrutural, consistência terminológica e aderência ao domínio e aos requisitos levantados.

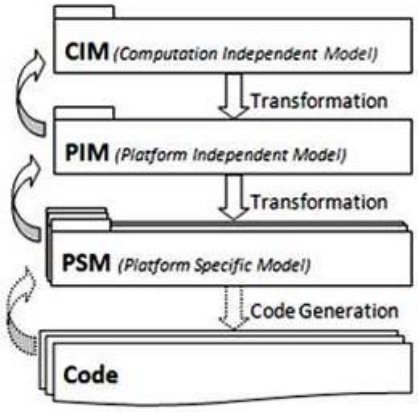
## 4. FUNDAMENTAÇÃO TEÓRICA

A crescente complexidade dos sistemas de software contemporâneos exige abordagens de desenvolvimento que promovam abstração, reusabilidade e adaptabilidade a diferentes plataformas tecnológicas (KLEPPE; WARMER; BAST, 2003). Nesse contexto, a *Model Driven Architecture* (MDA), definida pela *Object Management Group* (OMG), apresenta-se como uma abordagem orientada a modelos que visa estruturar o desenvolvimento de software de forma mais organizada e gerenciável.

A proposta da MDA baseia-se no uso de modelos como elementos centrais do processo de desenvolvimento, permitindo representar desde os requisitos de negócio até os artefatos técnicos do sistema. Segundo Duby (2003), a adoção dessa abordagem contribui para o aumento da produtividade e da qualidade do software ao automatizar tarefas repetitivas e reduzir a dependência de codificação manual. Além disso, a separação entre os modelos favorece a adaptação do sistema a novas tecnologias e requisitos, sem a necessidade de reestruturações extensas.

Na arquitetura MDA, o desenvolvimento é organizado em diferentes níveis de abstração. O *Computation Independent Model* (CIM) representa os requisitos do sistema e o entendimento do domínio do negócio, sem considerar aspectos computacionais. O *Platform Independent Model* (PIM) descreve a estrutura e o comportamento do sistema de forma independente de tecnologia, enquanto o *Platform Specific Model* (PSM) incorpora os detalhes necessários para a implementação em uma plataforma específica (OMG, 2014). Essa separação permite maior reutilização dos modelos, redução de custos de manutenção e identificação antecipada de falhas de projeto.

Figura 1- Níveis da Model Driven Architecture (MDA)




Fonte: An MDA Method for Automatic Transformation of Models from CIM to PIM, Journal of Software Engineering and Applications (2011).

A transformação estruturada entre os níveis CIM, PIM e PSM possibilita que os requisitos sejam preservados ao longo do processo de desenvolvimento. Conforme destacado por Mellor (2004), essa característica é especialmente relevante em sistemas que demandam rastreabilidade, pois minimiza ambiguidades e facilita a geração consistente de artefatos de software a partir de modelos abstratos.

No nível CIM, os casos de uso desempenham papel fundamental na representação das funcionalidades esperadas do sistema e das interações entre usuários e o sistema. Os casos de uso descrevem fluxos de eventos que podem ser organizados em um fluxo principal e fluxos alternativos, permitindo compreender o comportamento esperado em diferentes cenários (GUEDES, 2008).

A formalização dos cenários de casos de uso por meio da gramática BNF contribui para reduzir ambiguidades na especificação dos requisitos. Conforme proposto por Leite (2017), a utilização de regras sintáticas bem definidas possibilita maior precisão, consistência e rastreabilidade, além de viabilizar a transformação sistemática desses cenários em artefatos do nível PIM.

A formalização de cenários de caso de uso segundo uma gramática BNF apresenta desafios significativos. A linguagem natural utilizada na elicitação de requisitos tende a ser ambígua, incompleta e pouco padronizada, o que dificulta a

criação de regras sintáticas precisas capazes de representar adequadamente fluxos de eventos, atores e exceções. A aplicação manual da BNF nesse contexto exige elevado esforço de padronização textual, múltiplas revisões e conhecimento especializado em linguagens formais, tornando o processo lento e suscetível a erros.

Nesse contexto, a utilização de Large Language Models (LLMs), combinada com técnicas de engenharia de prompt, permite automatizar o refinamento dos cenários de caso de uso, convertendo descrições informais em textos sintaticamente consistentes com a gramática BNF adotada. A engenharia de prompt consiste na definição explícita de instruções, restrições sintáticas e formatos de saída para orientar o comportamento de modelos de linguagem em tarefas específicas. Segundo Liu et al. (2023), estratégias de prompting possibilitam controlar a estrutura, o nível de detalhe e a consistência das respostas geradas por LLMs, tornando-as adequadas a contextos formais e a processos de transformação automática. No presente trabalho, tais estratégias são aplicadas para restringir a geração textual ao padrão canônico Sujeito–Verbo–Objeto, assegurando aderência às regras da instância de Model Driven Architecture e preparando os cenários do nível CIM para transformação sistemática em modelos UML no nível PIM.

A partir dos cenários formalizados, podem ser gerados modelos estruturais e comportamentais, com destaque para o diagrama de classes e o diagrama de sequência. O diagrama de classes representa as classes do sistema, seus atributos, métodos e relacionamentos, fornecendo uma visão estática da estrutura e servindo como base para o projeto orientado a objetos (LARMAN, 2002). O diagrama de sequência, por sua vez, evidencia a interação entre os objetos ao longo do tempo, destacando a ordem das mensagens trocadas durante a execução de uma funcionalidade específica (GUEDES, 2008).

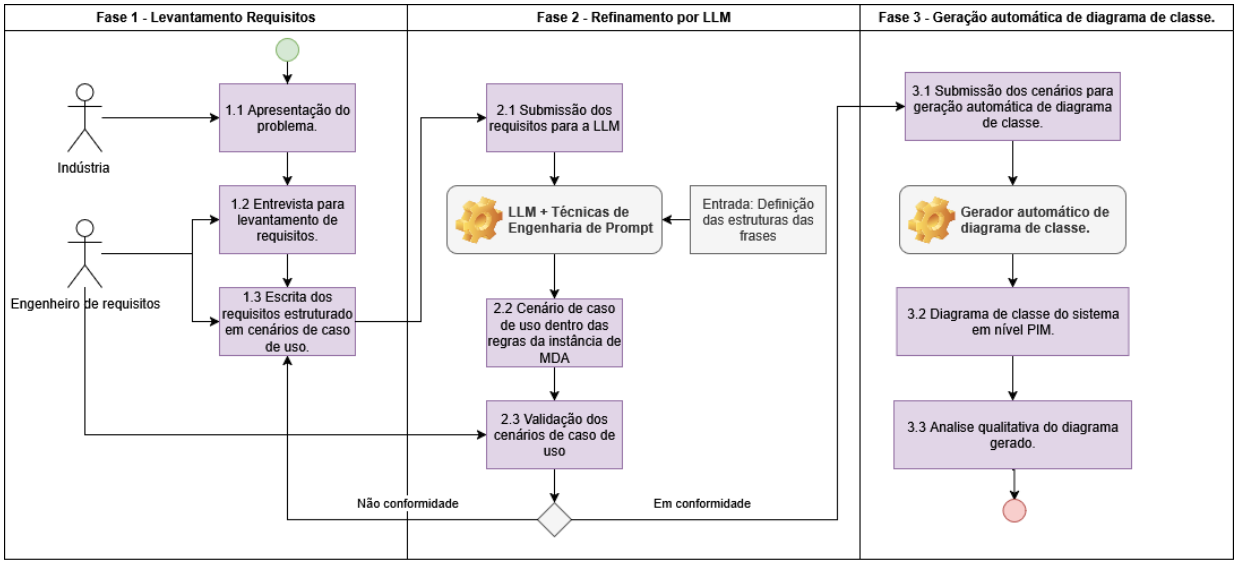
A organização dos modelos gerados pode ser apoiada pela arquitetura Model-View-Controller (MVC), que propõe a separação entre lógica de negócio, controle e interface. Essa separação contribui para a clareza estrutural do sistema e facilita sua manutenção e evolução, permitindo que alterações sejam realizadas de forma

localizada, sem comprometer o entendimento global da aplicação (ISIKDAG; UNDERWOOD, 2010).

## 5. METODOLOGIA

A metodologia organiza a transformação de requisitos em modelos UML em três fases integradas, com artefatos intermediários controlados e um ciclo explícito de conformidade para reduzir ambiguidades e garantir que a entrada da etapa automática seja suficientemente estruturada. Observa-se o fluxo metodológico na Figura 2 que parte de requisitos obtidos em contexto real fornecido pela indústria e evolui para uma representação textual padronizada, adequada à aplicação de regras de transformação no contexto de MDA, culminando na geração de um diagrama de classes em nível PIM e sua avaliação.

Figura 2- Fluxo metodológico em três fases para transformar requisitos em cenários de caso de uso e gerar automaticamente o diagrama de classes (PIM).




Fonte: Elaborado pelo autor (2025).

Na Fase 1, dedicada ao levantamento de requisitos, a industria realiza a apresentação do problema (1.1), caracterizando o contexto do domínio, os objetivos do sistema e restrições relevantes. Em seguida, o engenheiro de requisitos conduz

entrevista para levantamento de requisitos (1.2), consolidando necessidades funcionais, regras de negócio, exceções e variações do processo. Como resultado, procede-se à escrita dos requisitos estruturado em cenários de caso de uso (1.3), produzindo um artefato intermediário com formato padronizado, capaz de explicitar elementos essenciais para posterior modelagem, como atores, fluxo principal e fluxos alternativos ou de exceção. Essa estruturação eleva a rastreabilidade entre o que foi elicitado e o que será derivado no modelo, ao mesmo tempo em que impõe um nível mínimo de completude e organização ao texto.

Na Fase 2, o refinamento por LLM é empregado como mecanismo de normalização dos cenários segundo as regras de BNF que define a estrutura sintática das frases dos cenários de caso de uso. Inicialmente ocorre a submissão dos requisitos para a LLM (2.1), combinando técnicas de engenharia de prompt com a entrada referente à definição das estruturas das frases, de modo a restringir o formato e o nível de detalhamento esperado na saída. A etapa seguinte produz o cenário de caso de uso dentro das regras da instância de MDA (2.2), no qual ações são descritas de forma a atender as características sintática imposta pela instância de MDA descrita. Essa padronização favorece a transformabilidade do texto e o mapeamento posterior para classes, atributos e relacionamentos. Em seguida, realiza-se a validação dos cenários de caso de uso (2.3), caracterizada como um ponto de controle de conformidade que verifica consistência interna, completude e aderência ao padrão definido. Quando a validação identifica não conformidade, o fluxo retorna para ajustes no artefato, realimentando a etapa de estruturação/refinamento até que o cenário esteja em conformidade, quando então o processo avança para a fase de geração automática.

Na Fase 3, ocorre a submissão dos cenários para geração automática de diagrama de classe (3.1), etapa na qual os cenários validados são fornecidos ao gerador automático de diagrama de classe. O gerador aplica regras de transformação para produzir o diagrama de classe do sistema em nível PIM (3.2), preservando a independência de plataforma e a separação entre especificação conceitual e implementação. A partir da interpretação estruturada dos passos do cenário refinado,

o mecanismo identifica candidatos a classes do domínio, atributos e associações, derivando responsabilidades ou operações quando aplicável. Por fim, a metodologia prevê a análise qualitativa do diagrama gerado (3.3), verificando coerência estrutural, consistência terminológica e aderência às regras de negócio expressas nos cenários. Essa análise fornece evidências sobre a adequação do modelo gerado e subsidia o aprimoramento contínuo das estruturas frasais, das estratégias de prompt e das regras do gerador, reforçando a robustez metodológica e a reprodutibilidade do processo em diferentes instâncias de aplicação.

## 6. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Esta seção apresenta e discute os resultados obtidos a partir da aplicação do sistema proposto em um contexto industrial real, por meio de um estudo de caso conduzido na indústria Ucelo. O foco da análise está na validação prática da abordagem baseada em *Model Driven Architecture* (MDA), avaliando a capacidade do sistema em transformar requisitos informais em modelos UML no nível *Platform Independent Model* (PIM), a partir de artefatos do nível *Computation Independent Model* (CIM).

Os resultados são apresentados de maneira sequencial, seguindo o fluxo metodológico deste trabalho: começamos contextualizando o problema da empresa, depois definimos os casos de uso em linguagem natural, aplicamos a *Large Language Model* (LLM) para aprimorar a clareza e o sentido dos textos, e, por fim, geramos os modelos UML correspondentes.

### 6.1 Contextualização do problema empresarial

A indústria Ucelo atua no desenvolvimento de soluções industriais para transporte vertical de grãos e produtos granulados, sendo especializada no projeto e dimensionamento de elevadores de canecas. Atualmente, o processo de dimensionamento técnico é realizado majoritariamente por meio de planilhas complexas (Figura 3), desenvolvidas internamente ao longo dos anos.

Essas planilhas reúnem um grande volume de regras de negócio, fórmulas matemáticas, parâmetros normativos e critérios técnicos, de modo que seu correto preenchimento e interpretação dependem fortemente da experiência do engenheiro responsável. Embora cumpram adequadamente sua função operacional, o estudo identificou limitações relevantes, tais como a dificuldade de manutenção e evolução das planilhas ao longo do tempo, a baixa rastreabilidade entre requisitos, cálculos e decisões de projeto, a forte dependência de conhecimento tácito dos especialistas envolvidos, a ausência de modelos formais que apoiem processos de validação, reutilização e automação, bem como a limitada integração com sistemas de apoio à decisão no contexto comercial.

Esse cenário motivou a investigação da viabilidade de extrair e formalizar os requisitos implícitos nas planilhas, transformando-os em modelos estruturados de engenharia de software por meio da abordagem MDA.

Figura 2 - Planilha no Excel utilizada para comparação de canecas.

**COMPARATIVO DE CANECAS**

| CANECA SELECIONADA | | | | | | | | CANECA PARA COMPARAÇÃO | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Velocidade | 3,03 | | | m/s | | | | Velocidade | 3,00 | | | m/s |
| Densidade do produto | 750 | | | kg/m³ | | | | Densidade do produto | 750 | | | kg/m³ |
| Modelo caneca | 18X8M5 | | | | | | | Modelo caneca | 11X6 | | | |
| **Dimensões** | **479** | **197** | **146** | ***mm*** | 190 | 20 | 25 | mm | **Dimensões** | **289** | **177** | **121** | ***mm*** |
| **Volume da caneca** | **7,8** | | | ***l*** | 5,22 | | (33,1%) | l | **Volume da caneca** | **2,58** | | | ***l*** |
| Nº de fileiras | | | | | | | | | Nº de fileiras | 1 | | | |
| Passo | 156 | | | mm | | | | | Passo | 134 | | | mm |
| **Canecas/metro** | **0,00** | | | | -7,46 | | (0%) | | **Canecas/metro** | **7,46** | | | |
| **Furação** | **6F.D11.EF80.BF35** | | | | | | | | **Furação** | **3F.D9.EF110.BF30** | | | |
| Material | PEAD EAGLE | | | | | | | | Material | Nylon 6 | | | |
| Enchimento | 95% | | | | -5% | | (-95%) | | Enchimento | 100% | | | |
| **Capacidade** | **0,00** | | | ***t/h*** | -155,96 | | (0%) | t/h | **Capacidade** | **155,96** | | | ***t/h*** |
| **Volume de borda** | **0,00** | | | ***cm³*** | -95,35 | | (0%) | cm³ | **Volume de borda** | **95,35** | | | ***cm³*** |
| **Resistência a abrasão** | **0,00** | | | | -0,56 | | (0%) | | **Resistência a abrasão** | **0,56** | | | |
| **Resistência a tração** | **567,71** | | | ***N*** | -671,54 | | (-45,8%) | N | **Resistência a tração** | **1239,25** | | | ***N*** |
| **Deslocamento** | **11,59** | | | ***mm*** | 1,46 | | (87,4%) | mm | **Deslocamento** | **10,13** | | | ***mm*** |
| Preço/Unidade | | | | | -R$ | 45,00 | (0%) | | Preço/Unidade | R$ | 45,00 | | |
| **Preço/metro** | **R$** | - | | | -R$ | 335,82 | (0%) | | **Preço/metro** | **R$** | **335,82** | | |

Fonte: Elaborado pelo autor (2025).

A análise da planilha evidenciou que, embora os cálculos estejam corretos do ponto de vista técnico, a ausência de uma modelagem formal dificulta a compreensão global do processo e inviabiliza a automação direta para software.

### 6.2 Casos de uso iniciais no nível CIM

Para compreender integralmente o processo de dimensionamento, a coleta de informações combinou diferentes estratégias complementares. Inicialmente, foi realizada a análise das planilhas existentes, permitindo a identificação de fórmulas, parâmetros técnicos, dependências entre componentes e regras de validação. Em seguida, foram conduzidas sessões de esclarecimento com o representante técnico da empresa Ucelo, as quais possibilitaram validar interpretações, esclarecer cálculos e compreender as expectativas em relação a um sistema computacional. Complementarmente, realizou-se uma pesquisa documental, com consulta a manuais técnicos, fichas de produtos e informações públicas disponíveis no site da empresa.

A partir dessas atividades, foram identificados os principais processos envolvidos no dimensionamento, que inicialmente foram descritos em linguagem natural, de forma informal, refletindo a maneira como o domínio é compreendido pelos especialistas da empresa.

A seguir apresenta-se um exemplo de caso de uso descrito em seu formato inicial, ainda fora do padrão exigido pela instância MDA adotada:

“O usuário informa o nome do projeto, a altura de elevação, a velocidade do elevador, a densidade do produto e a capacidade desejada da caneca, e submete os dados ao sistema. Em seguida, o sistema processa as entradas e calcula a potência requerida, a potência recomendada (em cavalos-vapor) e o torque do motor. Por fim, o sistema determina o comprimento da caneca e calcula a sua capacidade resultante.”

Embora compreensíveis para humanos, descrições desse tipo apresentam ambiguidades, sujeitos implícitos, verbos genéricos e múltiplas ações em uma única frase, o que inviabiliza sua utilização direta em um processo automatizado de transformação de modelos no contexto da MDA.

### 6.3 Normalização sintática dos requisitos com base no padrão BNF

Com o objetivo de viabilizar a transformação dos requisitos informais em artefatos formais do nível Computation Independent Model (CIM), foi utilizado um agente baseado em Large Language Model (LLM), configurado para realizar a normalização sintática e semântica dos cenários de casos de uso. Esse agente, foi utilizado o gpt4, foi definido a partir de um prompt estruturado segundo a gramática BNF proposta por Leite (2017), garantindo rigor linguístico e aderência às regras da instância MDA adotada. O prompt completo utilizado para a normalização dos requisitos e geração do XML no nível CIM encontra-se descrito no Apêndice A.2.

A atuação desse agente seguiu rigorosamente o padrão canônico Sujeito–Verbo–Objeto (SVO), conforme definido na gramática BNF proposta por Leite (2017). Nesse padrão, cada sentença deve apresentar explicitamente:

um sujeito, que pode representar um ator externo (por exemplo, Usuário) ou o próprio Sistema;

um verbo, expresso no presente do indicativo, representando uma ação ou evento;

um objeto, correspondente à entidade principal do domínio, cuja última palavra define a classe a ser derivada;

um complemento, apenas quando semanticamente necessário e permitido pela gramática.

A adoção explícita do padrão é fundamental para a abordagem MDA, pois cada elemento da sentença possui papel direto na derivação dos modelos UML. O sujeito define o tipo de interação ou processamento, o verbo representa uma operação ou evento, e o objeto determina a classe principal no diagrama de classes. Dessa forma, as sentenças normalizadas passam a constituir formalmente o nível CIM, permitindo uma transformação determinística para o nível *Platform Independent Model* (PIM).

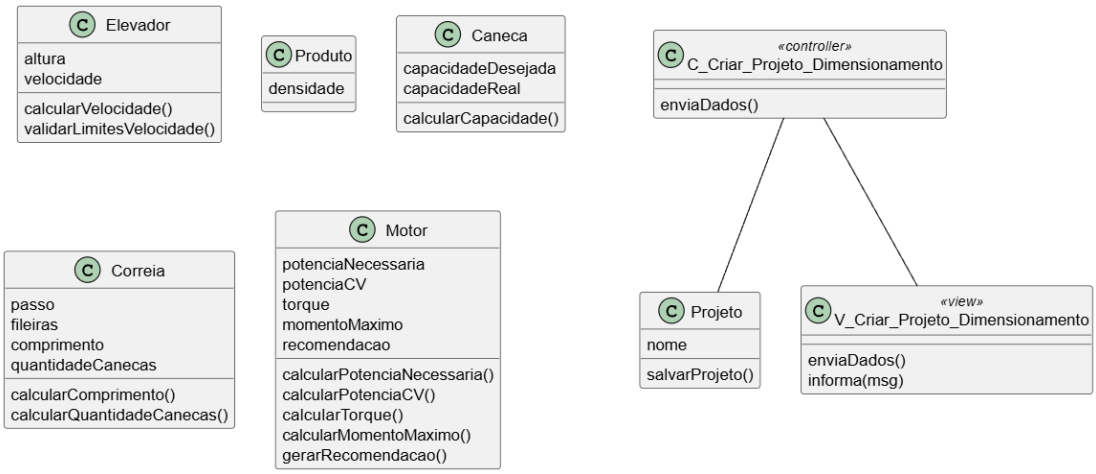
Qualquer violação desse padrão sintático compromete diretamente a transformação automática dos modelos e, consequentemente, a geração consistente dos diagramas UML e do código-fonte, reforçando a necessidade de rigor linguístico na definição dos cenários de casos de uso.

### 6.4 Geração dos cenários refinados e modelos UML no nível PIM

Após a aplicação do agente de normalização, os cenários de casos de uso passaram a apresentar estrutura sintática uniforme e aderente à gramática definida, eliminando ambiguidades e ruídos semânticos presentes nas descrições originais.

Esses cenários refinados foram posteriormente submetidos ao sistema de transformação, resultando na geração automática de modelos UML no nível *Platform Independent Model* (PIM), incluindo diagramas de classes e diagramas de sequência. O Apêndice A.1 apresenta um exemplo de cenário refinado, estruturado conforme a gramática adotada, evidenciando o formato de entrada empregado pelo mecanismo de geração automática. O diagrama de classes gerado encontra-se apresentado na figura 4, ilustrando os elementos estruturais. A análise dos artefatos gerados indicou coerência entre os requisitos normalizados, os elementos estruturais identificados e o domínio técnico da empresa Ucelo.

Figura 3 – Diagrama de classe no nível PIM gerando no PlantUML




Fonte: Elaborado pelo autor (2025).

Embora a transição para *Platform Specific Model* (PSM) e a geração de código não tenham sido o foco principal deste estudo, os resultados obtidos demonstram que os modelos gerados a partir do CIM normalizado estão aptos a suportar essas etapas em trabalhos futuros, validando a aplicabilidade da abordagem proposta em um ambiente empresarial real.

## 7. CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo conceber, implementar e avaliar uma metodologia baseada em Model Driven Architecture, apoiada por Large Language Models, para transformar requisitos funcionais textuais em artefatos formais e viabilizar a geração automática de diagramas de classes UML no nível Platform Independent Model (PIM). A partir dos resultados obtidos, é possível afirmar que os objetivos propostos foram alcançados.

A aplicação da abordagem em um estudo de caso real na indústria Ucelo permitiu evidenciar que a normalização sintática dos requisitos, conduzida por uma LLM orientada por regras explícitas de gramática BNF, contribui significativamente para a redução de ambiguidades e para o aumento da padronização dos cenários de caso de uso no nível CIM. Essa padronização mostrou-se fundamental para a aplicação das regras de transformação e para a geração consistente dos modelos UML no nível PIM.

Os resultados indicam que a utilização de LLMs como ferramenta de apoio à engenharia de requisitos, quando integrada a uma metodologia formal como a MDA, é viável e capaz de apoiar o processo de modelagem, especialmente em contextos nos quais os requisitos são inicialmente descritos de forma informal e dependem fortemente do conhecimento tácito dos especialistas do domínio. A geração automática do diagrama de classes apresentou coerência estrutural, consistência terminológica e aderência ao domínio analisado, demonstrando o potencial da abordagem como suporte à atividade de projeto de software.

Como trabalhos futuros, sugere-se a ampliação da metodologia para contemplar a transformação dos modelos gerados para o nível Platform Specific Model (PSM) e a geração automática de código-fonte, bem como a avaliação da abordagem em outros domínios industriais. Também se destaca a possibilidade de investigar métricas quantitativas para avaliar a qualidade dos modelos gerados e comparar o desempenho da abordagem proposta com métodos tradicionais de modelagem manual.

## REFERÊNCIAS

ABDELOUAHAB, Z.; EL AMRANI, M.; RACHIDI, T. An MDA method for automatic transformation of models from CIM to PIM. *Journal of Software Engineering and Applications*, v. 4, n. 1, p. 1–9, 2011.

DUBY, Carolyn K. Accelerating embedded software development with a Model Driven Architecture. Pathfinder Solutions, set. 2003. Disponível em: https://www.omg.org/mda/mda_files/MDA_overview.pdf. Acesso em: 26 jun. 2025.

GUEDES, G. T. *UML 2 – uma abordagem prática*. 1. ed. [S.l.]: [s.n.], 2008.

ISIKDAG, U.; UNDERWOOD, J. Two design patterns for facilitating building information model-based synchronous collaboration. *Automation in Construction*, Amsterdam, v. 19, n. 5, p. 544–553, 2010.

KLEPPE, Anneke; WARMER, Jos; BAST, Wim. *MDA explained: the model driven architecture: practice and promise*. Boston: Addison-Wesley, 2003.

LARMAN, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3. ed. Upper Saddle River: Prentice Hall, 2004.

LEITE, V. M. *Uma instância de uma arquitetura orientada a modelo e suas implicações na implentação dos processos do mps.br*. Dissertação (Mestrado) — Universidade Estadual de Londrina, Londrina, 2017.

LIU, P.; YUAN, W.; FU, J.; JIANG, Z.; HAYASHI, H.; NEUBIG, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 2023.

MELLOR, Stephen J. *MDA distilled: principles of model-driven architecture*. Boston: Addison-Wesley Professional, 2004.

OBJECT MANAGEMENT GROUP (OMG). *Model Driven Architecture (MDA) Guide*. Version 1.0. Needham, MA: Object Management Group, 2014.

OBJECT MANAGEMENT GROUP (OMG). *Model Driven Architecture (MDA) Guide*. Version 2.0. Needham, MA, 2014. Disponível em: https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf. Acesso em: 27 jun. 2025.

## APÊNDICE

Quadro A.1 – Código XML com cenário de caso de uso selecionado.

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:ext="urn:umlweb:v1">
  <use_case id="cc3ac154-618d-4369-9d27-c672d209288c">
    Criar Projeto de Dimensionamento
    <main_flow>
      <phrase id="f0673363-8589-4f00-83a3-a7b7a9be4911">Usuário insere nome do
Projeto</phrase>
      <phrase id="432043b0-23bd-4978-aa6c-7b20a4e0aadd">Usuário insere altura, velocidade
do Elevador</phrase>
      <phrase id="92afb95c-b4ee-477e-840d-7adf9597bcbe">Usuário insere densidade do
Produto</phrase>
      <phrase id="8528002c-e2ff-4dee-8816-61f9a4b81f11">Usuario insere capacidadeDesejada
da Caneca</phrase>
      <phrase id="cade3852-ec87-4bb1-815f-771c0347c7d6">Usuário insere passo, fileiras da
correia</phrase>
      <phrase id="1db2c1d9-3cad-44ba-b9b1-a2c75b982d88">Usuário envia dados</phrase>
      <phrase id="78e715e1-52d8-4f98-8146-6b1279c24c01">Sistema calcula potenciaNecessária
do motor</phrase>
      <phrase id="87cc6e74-4079-4928-9dbc-93c8a8f67e86">Sistema calcula recomendação do
motor</phrase>
      <phrase id="5b3c3486-68a7-40f2-bca2-8b44f9bf3dc6">Sistema calcula potenciaEmCv do
motor</phrase>
      <phrase id="f4a36e04-4fc3-4ea8-af78-2f7a80c8910e">Sistema calcula torque do
motor</phrase>
      <phrase id="468e34fd-8562-4cf7-8e6c-9d5e0ca88f79">Sistema calcula comprimento da
correia</phrase>
      <phrase id="769c3c8b-c3ae-4821-a8a4-bc80dbe74462">Sistema calcula quantidade de
canecas</phrase>
      <phrase id="6bd65e0a-be32-4ebd-866e-7c39201c932f">Sistema calcula capacidade da
caneca</phrase>
      <phrase id="e4fbeb36-4f95-491c-aa0e-f2a94dcfbcf5">Sistema calcula momentoMaximo do
motor</phrase>
    </main_flow>
    <alternative_flow id="a4f65f21-932b-4978-a0d3-2d80fd710837"
ext:parent_phrase_id="1db2c1d9-3cad-44ba-b9b1-a2c75b982d88" ext:kind="alternative">
      Caneca incompatível
      <flow id="2f03307e-1418-492a-943b-333cdc9842c6">Sistema INFORMA "Caneca incompatível
com projeção ou passo informado"</flow>
    </alternative_flow>
  </use_case>

  <ext:layout>
    <ext:position refType="node" refId="cc3ac154-618d-4369-9d27-c672d209288c" x="498"
y="172.9499969482422"/>
  </ext:layout>
  <ext:meta exportedBy="uml-web" version="1.0"/>
</root>
```

Fonte: Elaborado pelo autor (2025).

Apêndice A.2 - Prompt utilizado para normalização sintática com LLM

OBJETIVO DO AGENTE: Transformar requisitos informais ou mal escritos em sentenças sintaticamente normalizadas segundo Leite (2017), obedecendo às regras da gramática BNF utilizada na instância de MDA descrita no artigo. A normalização deve corrigir ambiguidades e garantir rigor sintático, pois estas sentenças constituem o nível CIM (Computation Independent Model) do método e serão automaticamente derivadas para o nível PIM (Platform Independent Model), gerando diagramas de classes, diagramas de sequência e, posteriormente, código-fonte Java. O XML produzido por este agente será utilizado diretamente no processo MDA; portanto, qualquer erro de verbo, objeto, estrutura, tag, sintaxe ou nome compromete a transformação do modelo e a geração correta do código.

BASE TEÓRICA UTILIZADA:

Segundo Leite (2017), a instância de MDA exige rigor sintático nas frases do cenário de caso de uso, pois a transformação CIM → PIM ocorre por regras formais estabelecidas em uma gramática (BNF). As frases normalizadas permitem derivar, de forma determinística, artefatos estruturais e comportamentais, resultando em diagramas consistentes e código-fonte coerente com os modelos. Como o CIM é a base das transformações subsequentes (CIM → PIM → PSM → Código), a precisão linguística é mandatória.

ENTRADA: O usuário fornecerá uma frase informal representando parte de um caso de uso. A frase pode conter sujeito omitido, verbo inadequado, objeto mal-formado, ambiguidade, informalidade, excesso de preposições, ruído semântico e imprecisões linguísticas.

CORREÇÃO SEMÂNTICA (converter o que veio no que deve virar):

Correção do sujeito: Se implícito, escolher entre Usuário, Ator relevante ou Sistema. O sujeito deve ser explícito para permitir derivação formal.

Correção do verbo: Converter sempre ao presente do indicativo, com verbos preferenciais definidos pela metodologia (ex.: insere, informa).

Correção do objeto: Identificar a entidade principal do domínio e reescrever o

objeto de forma adequada, com a última palavra representando a entidade dominante (ex.: “solicitações de reembolso feitas pelos usuários” → “solicitação de reembolso”).

Correção do complemento: Usar apenas quando fizer sentido conceitual, respeitando o formato da sintaxe da gramática BNF descrita pelo método.

Eliminar toda verbosidade, ruído e construções irrelevantes ao CIM.

REGRAS DE NORMALIZAÇÃO (baseadas na gramática BNF de Leite):

A frase normalizada deve seguir o padrão:

<Sujeito> <Verbo> <Objeto> [<Complemento>]

Devem ser evitados: artigos, pronomes, preposições desnecessárias ou estruturas que dificultem a derivação CIM → PIM. A última palavra deve ser a entidade principal (ponto essencial para derivação do diagrama de classes).

DENTIFICAÇÃO (RFXX):

Cada sentença deve receber um identificador:

RFXX - <Sujeito> <Verbo> <Objeto> [<Complemento>]

O número deve ser sequencial.

REGRAS DE DERIVAÇÃO (explicitamente reforçadas pelo artigo):

Como o XML será usado como insumo formal da transformação CIM → PIM → PSM → Código, obedecer estritamente:

Se o Sujeito for um ator (Usuário, Administrador, Cliente etc.), derivar classes V_<CasoDeUso> e C_<CasoDeUso>. O verbo representa um evento de interface.

Se o Sujeito for Sistema, derivar um método interno na classe do objeto.

Essas regras correspondem diretamente à etapa de derivação CIM para PIM descrita no estudo, incluindo os diagramas comportamentais (sequência) e estruturais (classes).

XML QUE O AGENTE DEVE PRODUZIR (estrutura formal utilizada pela instância MDA):

O XML deve ser gerado exatamente neste padrão, pois será interpretado por ferramentas automáticas de transformação de modelos e geração de código:

<?xml version='1.0' encoding='ISO-8859-1'?> <root> <use_case id="X">Nome do Caso de Uso <main_flow> <phase id="0"> RFXX - <Sujeito> <Verbo> <Objeto>

[<Complemento>] <noun>Sujeito</noun> <verb>Verbo</verb> <object>Objeto</object> <complement>Complemento</complement> </phase> </main_flow> </use_case> </root>

A tag <complement> deve ser removida se não houver complemento. IDs devem ser incrementados automaticamente. A estrutura deve estar rigorosamente correta porque será usada para gerar automaticamente:

diagrama de classes

diagrama de sequência

código-fonte Java

conforme o fluxo do MDA descrito no artigo.

FLUXOS ALTERNATIVOS:

Se a frase representar erro, exceção ou condição alternativa, gerar:

<alternative_flow id="Y">

<flow id="0">Sistema informa <mensagem></flow>

</alternative_flow>

EXEMPLO CORRETO DO PADRÃO:

Entrada: “Quando o administrador estiver logado, ele deve ter a possibilidade de aprovar as solicitações de reembolso feitas pelos usuários.”

Saída:

<?xml version='1.0' encoding='ISO-8859-1'?> <root> <use_case id="5">Aprovar Solicitação de Reembolso <main_flow> <phase id="0"> RF05 - Administrador aprova solicitação de reembolso <noun>Administrador</noun> <verb>aprova</verb> <object>solicitação de reembolso</object> </phase> </main_flow> </use_case> </root>

MODO TESTE:

Se o usuário solicitar “gerar testes”, o agente deverá produzir 10 frases propositalmente inadequadas e gerar para cada uma:

frase normalizada conforme a gramática utilizada na instância

XML correto

garantindo conformidade com o processo CIM → PIM → PSM → Código.

INSTRUÇÃO FINAL DO AGENTE:

Gerar sempre frases sintaticamente corretas, semanticamente consistentes, seguindo a gramática BNF da metodologia, garantindo que cada sentença permita derivação automática para diagramas comportamentais e estruturais, e por fim código-fonte Java. Nunca inventar tags. Nunca produzir XML fora do padrão. Nunca ignorar regras de sintaxe, pois todo o XML será usado em transformações automáticas da instância MDA.


[Recebido/Received: Abril 30, 2023; Aceito/Accepted: Agosto 29, 2023]