APLICAÇÃO DE REGRAS PARA TRANSFORMAÇÃO DE MODELO INDEPENDENTE DE COMPUTAÇÃO (CIM) PARA MODELO INDEPENDENTE DE PLATAFORMA (PIM): um Estudo de Caso aplicado na Indústria Vidraceira

Flávio Henrique de Oliveira*, Anthon Pedrollo Hax, Maria Sophia Pardinho Da Silva, Pedro Delgado Henriques e Victor Shoiti Satake

RESUMO

Este estudo examina a aplicabilidade de uma instância da Model Driven Architecture (MDA) no desenvolvimento de um sistema de monitoramento da produção industrial para a empresa Vidraceira Temperlândia. A pesquisa tem como objetivo avaliar a capacidade da abordagem MDA em viabilizar a derivação automatizada de artefatos estruturais e comportamentais a partir da modelagem de requisitos expressos em linguagem natural. O processo metodológico envolveu a elicitação de requisitos no ambiente organizacional, resultando na produção de artefatos no nível Computation Independent Model (CIM), tais como cenários de caso de uso descritos em uma gramática BNF proposta, diagramas de caso de uso e protótipos de interface gráfica. Posteriormente, foram aplicadas regras formais de transformação, com base no arcabouço proposto por LEITE (2017), para a geração de modelos no nível Platform Independent Model (PIM), especificamente diagramas de classes e de sequência. Os resultados obtidos evidenciam que a abordagem MDA contribui significativamente para a mitigação de ambiguidades semânticas, para a padronização dos artefatos e para a sistematização do processo de desenvolvimento orientado a modelos. Conclui-se que a instância de MDA investigada apresenta elevado potencial para suportar a transformação de especificações dos requisitos de software em representações formais compatíveis com processos de automação do desenvolvimento de software.

Palavras-chave: Model Driven Architecture; Engenharia de Software; Transformação CIM-PIM.

* Autor correspondente (e-mail): flavio.oliveira@sistemafiep.org.br

Application of Rules for the Transformation from Computation Independent Model (CIM) to Platform Independent Model (PIM): A Case Study Applied in the Glass Industry

ABSTRACT

This study empirically examines the applicability of an instance of Model Driven Architecture (MDA) in the development of an industrial production monitoring system for the company Vidraceira Temperlândia. The research aims to assess the ability of the MDA approach to enable the automated derivation of structural and behavioral artifacts from requirements modeling expressed in natural language. The methodological process involved the elicitation of requirements within the organizational environment, resulting in the production of artifacts at the Computation Independent Model (CIM) level, such as use case scenarios described using a proposed BNF grammar, use case diagrams, and graphical user interface prototypes. Subsequently, formal transformation rules—based on the framework proposed by LEITE (2017), were applied to generate models at the Platform Independent Model (PIM) level, specifically class and sequence diagrams. The results obtained demonstrate that the MDA approach significantly contributes to mitigating semantic ambiguities, standardizing artifacts, and systematizing the model-driven development process. It is concluded that the investigated MDA instance shows strong potential to support the transformation of software requirement specifications into formal representations compatible with software development automation processes.

Key words: Model Driven Architecture; Software Engineering; CIM-to-PIM Transformation.

1. INTRODUÇÃO

A crescente complexidade dos sistemas de software contemporâneos exige abordagens de desenvolvimento que promovam a abstração, a reusabilidade e a adaptabilidade a diferentes plataformas tecnológicas (Kleppe, Warmer e Bast, 2003).

Nesse contexto, a Model Driven Architecture (MDA), preconizada pela Object Management Group (OMG), emerge como uma estratégia robusta para guiar o ciclo de vida do software por meio de modelos, garantindo uma estrutura compreensível e gerenciável. Segundo Duby (2003), o uso do MDA permite aumentar a produtividade e a qualidade do desenvolvimento de software, ao automatizar tarefas repetitivas e reduzir a necessidade de codificação manual. A separação entre o modelo independente de plataforma (PIM) e o modelo específico de plataforma (PSM) facilita a adaptação do sistema a novas tecnologias e requisitos, sem que seja necessário reestruturar toda a aplicação. Essa abordagem também contribui para a reutilização de modelos em diferentes contextos, reduz os custos de manutenção e documentação, e permite detectar falhas de projeto nas fases iniciais, promovendo entregas mais ágeis e sistemas mais robustos.

Este artigo técnico-científico investiga a aplicabilidade empírica de uma instância de MDA proposta por (LEITE, 2017) em um projeto de desenvolvimento de software, especificamente para a otimização de processos na Indústria Vidraceira Temperlândia, com os seguintes objetivos específicos: verificar a viabilidade de aplicar a abordagem em um ambiente real; avaliar sua contribuição na sistematização e automação da modelagem de processos; identificar os benefícios e limitações percebidos no uso da arquitetura orientada a modelos; e analisar o potencial da técnica para gerar artefatos automatizados a partir de modelos, promovendo eficiência e reuso no desenvolvimento de software.

O cenário industrial contemporâneo, exemplificado pela Temperlândia, fundada em 1999 e operante nos segmentos moveleiro, de decoração e produção de vidros temperados, apresenta desafios intrínsecos à gestão de sistemas produtivos de alto volume. A análise preliminar, realizada mediante visita técnica à sede da

empresa, identificou uma lacuna crítica no sistema de gerenciamento de dados produtivos existente: a insuficiência no rastreamento e monitoramento de peças ao longo da linha de produção. Tal deficiência resulta em gargalos operacionais, incremento de retrabalho e comprometimento da eficiência global do processo. O monitoramento reativo de equipamentos, ativado apenas na entrada de uma peça na máquina e frequentemente resultando em inatividade após a conclusão do processo evidencia uma oportunidade substancial para a otimização proativa do fluxo produtivo.

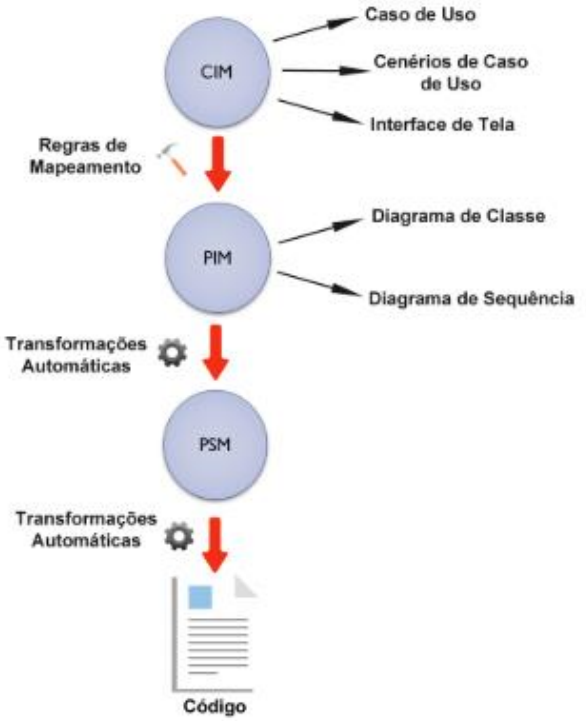
Diante do exposto, o MDA configura-se como uma abordagem metodológica promissora para abordar a supracitada lacuna tecnológica. Ao priorizar a construção de CIM – que representam os requisitos de negócio e as soluções em um nível de alta abstração – é possível derivar PIM, os quais orientam o design de sistemas desalinhados de especificidades tecnológicas. A transformação estruturada entre esses modelos assegura a incorporação dos requisitos de rastreabilidade desde as fases iniciais do projeto, minimizando ambiguidades e facilitando a geração de artefatos de software consistentes (MELLOR, 2004).

A aplicação desta metodologia demonstrou a viabilidade da geração automática de documentos comportamentais e estruturais de software, utilizando a instância de Leite (2017), como base para a transformação de modelos abstratos em concretos, por meio de uma arquitetura MVC (Model-View-Controller). Verificou-se que a ambiguidade e a duplicidade de funcionalidades entre classes podem ser significativamente reduzidas com o emprego de regras de transformação bem definidas. Este estudo evidencia a possibilidade de construir modelos estruturais e dinâmicos diretamente a partir de requisitos expressos em linguagem natural, apesar dos desafios inerentes à formalização dos cenários de caso de uso conforme as regras propostas.

2. FUNDAMENTAÇÃO TEÓRICA

O Model Driven Architecture (MDA), também conhecida como Arquitetura Dirigida por Modelos, é uma abordagem estruturada para o desenvolvimento de software definida e mantida pelo Object Management Group (OMG). Esse framework propõe a utilização de modelos como elementos centrais do processo de desenvolvimento, permitindo a representação desde os requisitos de negócio até a implementação final em plataformas tecnológicas. A adoção do MDA proporciona maior capacidade de lidar com a complexidade dos sistemas de grande porte, além de favorecer a interoperabilidade e a colaboração entre diversos atores e componentes do ecossistema computacional, incluindo pessoas, organizações, hardware e software (Object Management Group, 2014). Pode-se observar na Figura 1 as fases do MDA, destacando os artefatos empregados em cada etapa do processo.

Figura 1- Fases do MDA e seus artefatos



Fonte: Leite (2017).

As regras da sintaxe BNF (Backus-Naur Form) são utilizadas para a criação das frases dos cenários que especificam os casos de uso, exigindo que as frases dos cenários estejam de acordo com essa sintaxe. Na etapa PIM (Platform Independent Model), são definidos os diagramas de classe e de sequência, os quais são gerados a partir da derivação do diagrama de caso de uso e da interface de tela, seguindo regras estabelecidas com base na gramática BNF desenvolvida no estudo (LEITE, 2017).

Na arquitetura dirigida por modelos (MDA), os sistemas são descritos em diferentes níveis de abstração. O Modelo Independente de Computação (CIM) representa a visão mais abstrata, concentrando-se nos requisitos do sistema e em seu ambiente, sem considerar questões de implementação. Já o Modelo Independente de Plataforma (PIM) descreve a lógica do sistema sem depender de uma plataforma específica, permitindo que o mesmo modelo seja reutilizado em diferentes tecnologias. Por fim, o Modelo Específico de Plataforma (PSM) incorpora detalhes técnicos necessários para a implementação em uma plataforma concreta, combinando a abstração do PIM com elementos específicos da tecnologia-alvo (OMG, 2014).

O Model Driven Architecture propõe a separação entre os aspectos funcionais e técnicos de um sistema, permitindo que a lógica de negócio seja capturada em modelos independentes de plataforma (PIM), os quais podem ser automaticamente transformados em modelos específicos de plataforma (PSM). Essa transformação parte de um modelo computacionalmente independente (CIM), passando pelo PIM, até alcançar implementações concretas, promovendo reuso, adaptabilidade e automatização no processo de desenvolvimento (DUBY, 2003).

O diagrama de caso de uso é amplamente empregado na fase de análise de requisitos por representar as funcionalidades esperadas do sistema. Ele contribui para a identificação e compreensão dos requisitos, permitindo especificar, visualizar e documentar os serviços desejados pelos usuários. Além disso, cada caso de uso descreve uma sequência de interações entre o ator e o sistema, denominada fluxo,

que pode ser composto por um fluxo principal e por fluxos alternativos (GUEDES, 2008).

O diagrama de classes, por ser um diagrama estrutural, tem a função de representar as classes, interfaces, seus atributos, métodos e os relacionamentos entre elas, servindo como base para o projeto orientado a objetos. Esse diagrama é considerado um dos mais relevantes da UML, pois fornece uma visão estática e detalhada da estrutura do sistema, sendo amplamente utilizado na modelagem de software (LARMAN, 2002).

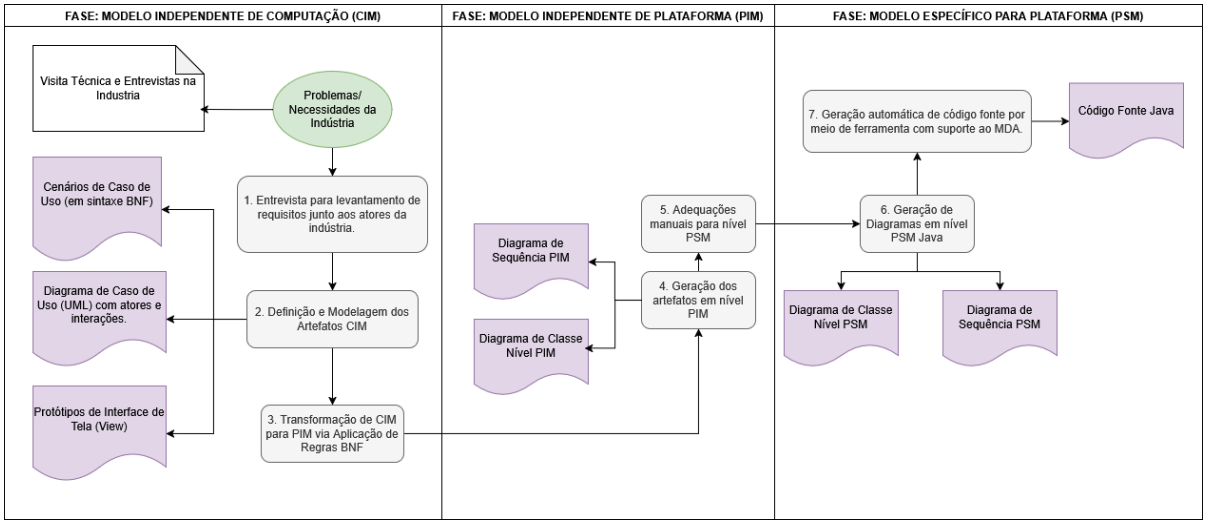
O diagrama de sequência é classificado como um diagrama comportamental da UML e tem como objetivo representar o comportamento dinâmico do sistema, destacando a interação entre os objetos ao longo do tempo. Sua principal característica é evidenciar a ordem temporal das mensagens trocadas entre os objetos participantes de um determinado processo, o que o torna essencial para a compreensão do fluxo de execução de funcionalidades do sistema (GUEDES, 2008).

A separação das responsabilidades entre os componentes de um sistema, como propõe a arquitetura MVC, contribui para a clareza estrutural e facilita a manutenção e evolução do software. Essa abordagem organiza o sistema em três camadas — *model, controller e view* —, promovendo a independência entre lógica de negócio, interface e controle, o que torna possível realizar alterações localizadas sem comprometer o entendimento global da aplicação (ISIKDAG; UNDERWOOD, 2010).

3. METODOLOGIA

A metodologia empregada neste projeto é de natureza aplicada, buscando analisar a aplicabilidade prática da instância de MDA em um contexto real da indústria. Especificamente, configura-se como um estudo de caso, ao investigar a fundo a aplicação da instância de MDA em um projeto desenvolvido para a Indústria Vidraceira Temperlândia. A Figura 2 é possível observar uma visão geral da abordagem proposta.

Figura 2 - Visão geral da abordagem proposta



Fonte: Elaboração própria.

A fase inicial do projeto concentrou-se na documentação dos requisitos de negócio de forma desvinculada de qualquer tecnologia computacional. Esta etapa foi crucial para compreender os desafios enfrentados pela Temperlândia, particularmente a ausência de um sistema eficaz de rastreamento e monitoramento de peças ao longo da linha de produção.

O levantamento de requisitos foi realizado por meio de uma visita técnica detalhada às instalações da empresa e entrevistas com especialistas técnicos de informática da empresa. Essa imersão permitiu identificar que, embora a empresa já utilizasse um sistema geral de gerenciamento de dados produtivos, havia uma necessidade premente de melhoria no processo de rastreamento de peças e no monitoramento de equipamentos. Problemas como gargalos operacionais, aumento de retrabalho e perda de eficiência foram diretamente atribuídos à falta de visibilidade contínua sobre o status e a posição dos itens no processo produtivo.

A partir da coleta de informações, foram construídos os principais artefatos do nível CIM, incluindo diagramas de casos de uso em UML, que representam as funcionalidades do sistema com a identificação clara dos atores e suas respectivas interações; cenários de casos de uso descritos em sintaxe BNF, seguindo rigorosamente as regras dessa notação conforme estabelecido na metodologia; e

protótipos de interfaces de tela, elaborados manualmente para cada caso de uso com interação de usuário, com o objetivo de garantir a rastreabilidade em relação aos casos de uso levantados e assegurar a usabilidade para os potenciais usuários.

A fase CIM, com seus artefatos detalhados, permitiu uma compreensão profunda dos processos internos da Temperlândia, identificando pontos críticos e guiando a proposta de uma solução de monitoramento integrada, essencial para a fase subsequente.

Esta é uma das etapas mais críticas e inovadoras da metodologia, pois aborda a lacuna na literatura sobre a transformação sistemática de CIM para PIM, um processo frequentemente realizado manualmente. A proposta metodológica utilizou regras de transformação baseadas na sintaxe BNF para especificação da UML, conforme expandido por Leite (2017). A ideia central é associar cada símbolo terminal da BNF (como sujeito, verbo, objeto e complemento) diretamente a elementos da UML, como classes de domínio, operações, atributos e relacionamentos.

Nesta fase de geração dos artefatos do nível PIM, o foco reside na modelagem do sistema de forma abstrata, ainda sem considerar a plataforma de implementação. Os artefatos PIM são derivados diretamente dos casos de uso definidos na etapa CIM. Os principais artefatos gerados incluem:

A transição de CIM para PIM, embora manual em alguns aspectos, foi sistemática e guiada pelas regras de transformação, o que garantiu a consistência dos modelos e permitiu a incorporação dos requisitos de rastreabilidade desde as fases iniciais do projeto

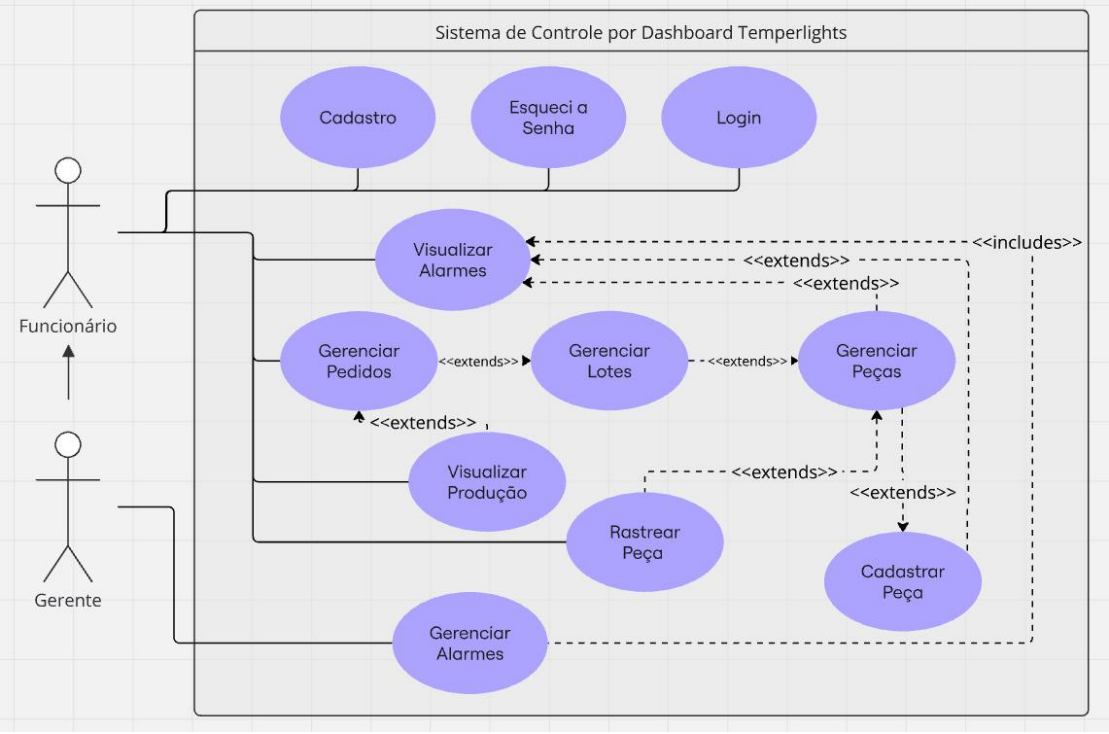
A etapa subsequente envolve as adequações para o PSM e a geração de código. Enquanto a transformação de PIM para PSM é mais amplamente discutida na literatura e suportada por ferramentas MDA que permitem sua automação, neste projeto, as adequações para o nível PSM foram consideradas, ainda que não detalhadas exaustivamente como as de CIM para PIM.

4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Nesta seção, será apresentado um cenários dos casos de uso e interfaces de tela correspondentes à fase CIM, bem como sua derivação no diagrama de classe e sequência referentes à fase PIM.

Na Figura 3 apresenta o diagrama de caso de uso consolidado, no qual estão representadas todas as funcionalidades mapeadas durante a fase de elicitação de requisitos.

Figura 3 - Resultado do diagrama de caso de uso completo.



Fonte: Elaboração própria.

Na Figura 4 apresenta o protótipo da interface desenvolvido para o caso de uso Gerenciar Lote. Nesta interface, o usuário pode visualizar diversos indicadores de desempenho, tais como a quantidade de peças em andamento, número de alertas, peças finalizadas e volume de perdas. Além disso, a tela exibe uma listagem detalhada do status individual de cada peça, destacando o tempo ocioso de cada uma, o que permite a identificação de potenciais perdas no processo produtivo.

Figura 4 - Protótipo de tela do caso de uso gerenciar lote.

Fonte: Elaboração própria.

Cada sentença do cenário do caso de uso foi estruturada de acordo com a sintaxe estabelecida pela metodologia adotada, possibilitando sua derivação sistemática para o diagrama de classes com base nas regras propostas por Leite (2017). Como ilustração da aplicação dessas regras, utiliza-se o fluxo principal do caso de uso Gerenciar Peça. A seguir, apresentam-se as sentenças extraídas, acompanhadas das respectivas regras aplicadas e suas derivações correspondentes.

Frase 1 – “Usuário insere código de barras da peça”. A análise da sentença permite identificar a presença do verbo “insere”, o qual caracteriza uma ação de entrada de dados por parte do usuário ou de um agente automatizado. De acordo com as regras de derivação e aprimoramento realizados, esta estrutura indica que a entidade Peça deve conter um atributo denominado código de barras. Assim, no diagrama de classes, é modelada a classe Peça, à qual é associado o atributo código de barras, representando a informação capturada pelo sistema por meio do usuário.

Frase 2 – “Usuário envia código de barras da peça”. A análise da sentença revela que o sujeito da ação “Usuário” difere de sistema, o que, segundo as regras de derivação estabelecidas, implica a criação de duas classes auxiliares: uma de interface (View) e uma de controle (Controller), ambas nomeadas com o prefixo do

caso de uso correspondente — neste caso, V_GerenciarPeça e C_GerenciarPeça, respectivamente. Em ambas as classes, é definido o método enviaCódigo(código), responsável por encapsular o dado transmitido.

No diagrama de sequência, a derivação implica na representação de duas mensagens: a primeira, originada no agente Usuário em direção à classe View, e a segunda, da View para a Controller, evidenciando o fluxo de dados dentro da arquitetura MVC.

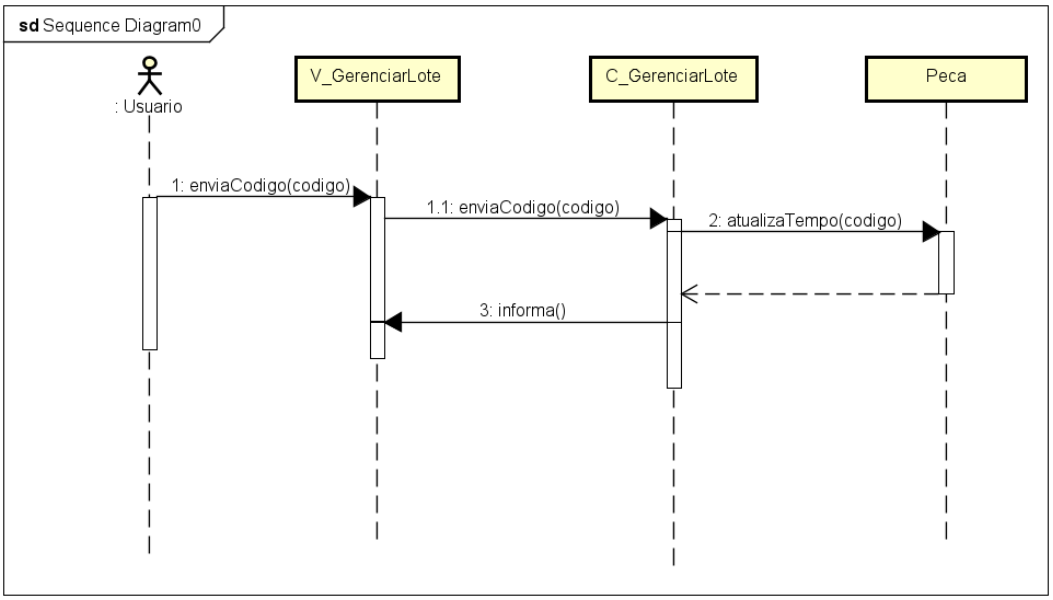
Frase 3 – “Sistema atualiza tempo inicial da peça”. A sentença segue a estrutura sintática do tipo: *<sujeito> <verbo> <objeto> <complemento 1> <complemento 2> <complemento 3>*. Ao analisar o sujeito, observa-se que ele corresponde ao próprio sistema, o que, conforme as diretrizes metodológicas, indica que a operação é realizada internamente pelo sistema sobre uma entidade do domínio. Dessa forma, no diagrama de classes, deriva-se a inclusão do método *atualizaTempo()* na classe *Peça*, representando a ação de atualização do atributo de tempo. No diagrama de sequência, a derivação implica em uma mensagem enviada da classe *Controller* para a classe *Peça*, invocando o método *atualizaTempo(codigo)*, evidenciando a lógica de controle exercida sobre a entidade do modelo.

Frase 4 – “Sistema informa código de peça”. A sentença segue a estrutura sintática <sujeito> <verbo> <objeto direto> <complemento 1> <complemento 2>, na qual o sujeito “Sistema” executa a ação. O verbo “informa” é classificado, conforme as regras de derivação propostas, como uma palavra reservada, indicando que há uma comunicação do Controller para a View.

Dessa forma, no diagrama de sequência, deriva-se uma mensagem originada na classe Controller em direção à classe View, por meio do método informa(msg), cujo parâmetro contém o conteúdo a ser exibido, neste caso, o código das peças.

O resultado pode-se observar na Figura 5 diagrama de sequência após derivações realizadas dos cenários de caso de uso de gerenciar lote.

Figura 5- Diagrama de sequência resultante da derivação do fluxo principal



Fonte: Elaboração própria.

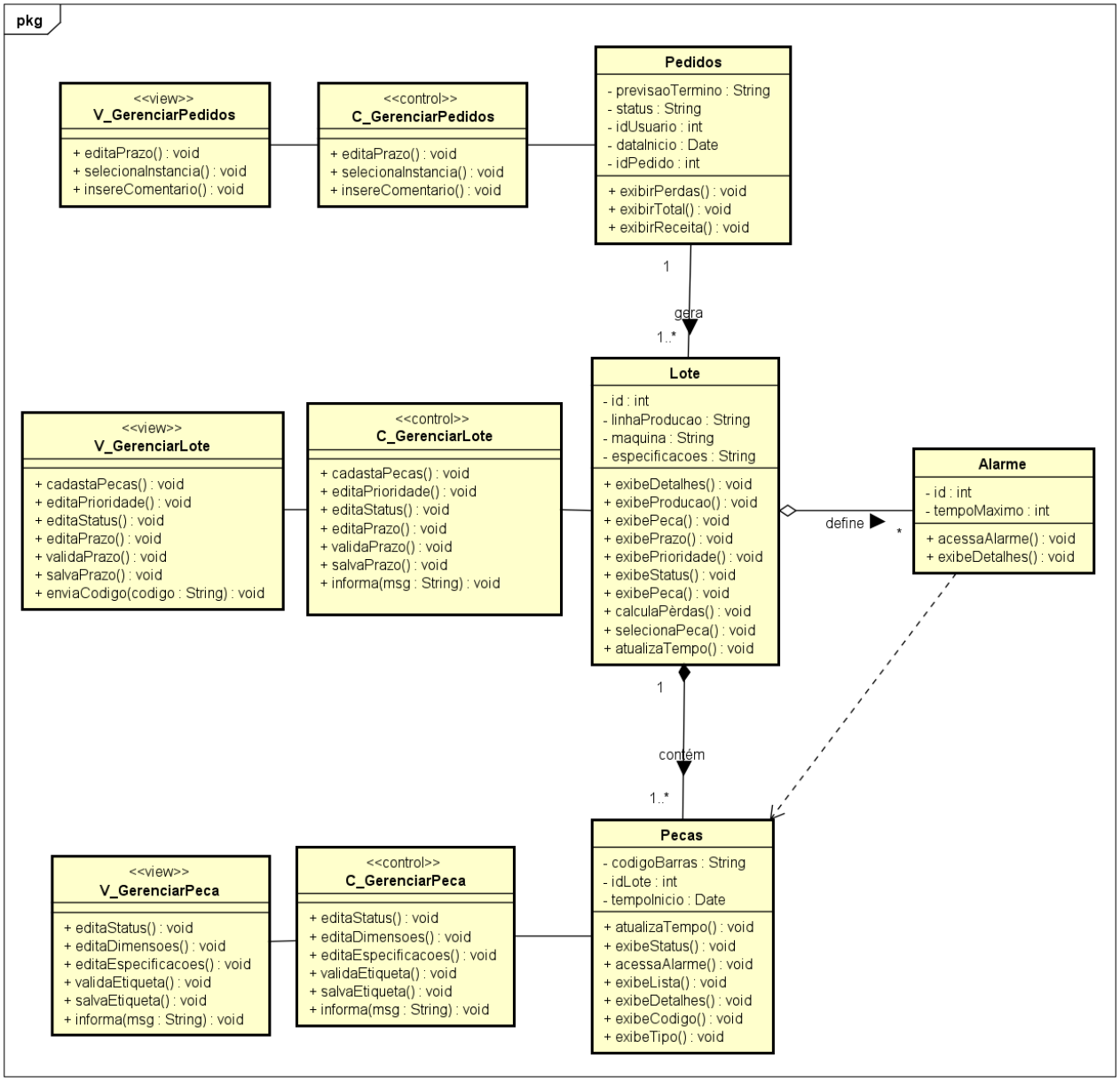
O modelo gerado pode, por meio da utilização de uma ferramenta com suporte à transformação automática do PSM para código fonte, viabilizar a geração automática do código na plataforma de destino desejada. Dessa forma, obtém-se um esqueleto de aplicação contendo as estruturas de classes e a definição dos métodos correspondentes, cabendo ao desenvolvedor a responsabilidade de implementar a lógica interna de cada método conforme os requisitos funcionais. Essa estrutura sistematizada promove um mapeamento direto entre os requisitos especificados, o modelo derivado e o código gerado, permitindo também a rastreabilidade bidirecional entre esses elementos ao longo do processo de desenvolvimento.

Na Figura 6 ilustra o diagrama de classes correspondente à modelagem completa do sistema, obtido a partir da derivação sistemática das sentenças extraídas dos cenários de casos de uso, por meio da aplicação das regras previamente definidas no processo de transformação. Após a geração inicial sistemática dos elementos estruturais, procedeu-se à realização de uma análise manual minuciosa, com o objetivo de estabelecer adequadamente os relacionamentos entre as classes identificadas, bem como definir de forma precisa as multiplicidades associadas a

essas relações, garantindo assim a coerência e a integridade semântica do modelo gerado.

O modelo gerado pode, por meio da utilização de uma ferramenta com suporte à transformação automática do Modelo Específico de Plataforma (PSM) para código fonte, viabilizar a geração automática do código na plataforma de destino desejada.

Figura 6 - Diagrama de classe completo



Fonte: Elaboração própria.

5. CONSIDERAÇÕES FINAIS

A utilização da instância metodológica proposta evidenciou-se eficaz na sistematização e automação parcial da modelagem de processos, permitindo a derivação consistente de artefatos estruturais e comportamentais a partir de requisitos expressos em linguagem natural.

A abordagem orientada a modelos contribuiu significativamente para a mitigação de ambiguidades semânticas, padronização de artefatos e rastreabilidade entre os elementos que compõem as etapas de análise, projeto e futura implementação. Dentre os benefícios observados, destacam-se a clareza na comunicação entre os stakeholders, a reutilização de modelos e a redução do esforço manual nas fases subsequentes do ciclo de desenvolvimento.

O processo sistematizado de transformação de modelos CIM para PIM demonstrou ser eficaz em promover consistência e alinhamento com os princípios da engenharia de software orientada a modelos. Nesse contexto, a construção de uma ferramenta de apoio que automatize integralmente esse processo mostra-se tecnicamente viável e desejável. Tal solução potencializaria os benefícios esperados da MDA, como a agilidade no desenvolvimento, o reuso de artefatos e a integração entre requisitos e implementação, consolidando-se como um recurso estratégico para ambientes industriais que demandam alta confiabilidade e produtividade.

Apesar dos avanços, identificaram-se limitações associadas à necessidade de formalização rigorosa dos cenários de caso de uso e à dependência de regras bem definidas para garantir a qualidade das transformações. Tais desafios apontam para a importância da capacitação técnica e da validação contínua durante a aplicação prática da metodologia. Nesse cenário, uma proposta promissora consiste na incorporação de soluções baseadas em *Large Language Models (LLMs)* para apoiar a adequação de textos em linguagem natural aos padrões exigidos pelas regras de transformação. A utilização de LLMs pode contribuir significativamente para a conversão automática e coerente de descrições informais em estruturas formais compatíveis com os modelos exigidos pela MDA, reduzindo o esforço humano e aumentando a aderência à metodologia.

REFERÊNCIAS

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 G.; WARMER, Jos B.; BAST, Wim. *MDA explained: the model driven architecture: practice and promise*. [S.l.]: Addison-Wesley Professional, 2003.

LEITE, V. M. Uma instância de uma arquitetura orientada a modelo e suas implicações na implantação dos processos do MPS.BR. 2017. Universidade Estadual de Londrina, Londrina, 2017.

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

OMG. *MDA Guide Version 2.0*. [S.l.], 2014. Disponível em: https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf. Acesso em: 27 jun. 2025.

[Recebido/Received: Dezembro 18 2024; Aceito/Accepted: Janeiro 29, 2025]