Revista Ibero-Americana de Humanidades, Ciências e Educação — REASE

doi.org/10.51891/rease.v12i3.25305

AUTOMAÇÃO DE MODELAGEM INICIAL EM BIM: UM ESTUDO DE CASO INTEGRANDO LINGUAGEM NATURAL E PYTHON

AUTOMATION OF EARLY-STAGE BIM MODELING: A CASE STUDY INTEGRATING NATURAL LANGUAGE AND PYTHON

AUTOMATIZACIÓN DE LA MODELACIÓN INICIAL EN BIM: UN ESTUDIO DE CASO QUE INTEGRA LENGUAJE NATURAL Y PYTHON

Daniel George Macêdo Neres[1]
Clara Maria Neres Macêdo[2]

RESUMO: Esse artigo buscou desenvolver um fluxo de automação para a modelagem inicial em *Building Information Modelling* (BIM), integrando linguagem natural e linguagem de programação. A metodologia consistiu em uma abordagem híbrida, onde um Modelo de Linguagem Grande (LLM) atua como interpretador semântico para estruturar dados em formato JSON, enquanto um algoritmo em *IronPython* opera como "auditor" e modelador na API do Autodesk Revit. Esse motor em *Python* aplica rigorosas validações geométricas, como cálculos de perímetro restrito, filtros de tolerância de curva e controle de cotas de elevação. Os resultados demonstraram que delegar o controle topológico exclusivo à Inteligência Artificial resulta em alucinações matemáticas e falhas estruturais, como a extrapolação de pilares transpassando pavimentos. Contudo, a intervenção do código Python mitigou 100% dessas inconsistências, permitindo a geração autônoma e precisa de esqueletos estruturais, beirais dinâmicos e aberturas conceituais. Conclui-se que o método de Auditoria Paramétrica Híbrida permite a tradução confiável de conceitos de modelagem, expressos em linguagem natural, em modelos paramétricos consistentes e geometricamente íntegros. A abordagem demonstrou potencial para otimizar a etapa de Estudo Preliminar, ao mesmo tempo em que supera limitações geométricas inerentes aos modelos de inteligência artificial generativa, garantindo maior estabilidade na geração automatizada de modelos BIM.

Palavras-chave: BIM. Modelagem Paramétrica. Inteligência Artificial.

ABSTRACT: This article aimed to develop an automation workflow for initial modeling in Building Information Modeling (BIM) by integrating natural language and programming language. The methodology consisted of a hybrid approach in which a Large Language Model (LLM) acts as a semantic interpreter to structure data in JSON format, while an IronPython algorithm operates as both an “auditor” and model generator through the Autodesk Revit API. This Python engine applies rigorous geometric validations, including restricted perimeter calculations, curve tolerance filters, and elevation constraint controls. The results demonstrated that delegating exclusive topological control to Artificial Intelligence leads to mathematical hallucinations and structural inconsistencies, such as pillar extrapolation across building floors. However, the intervention of the Python code mitigated 100% of these inconsistencies, enabling the autonomous and precise generation of structural skeletons, dynamic overhangs, and conceptual openings. It is concluded that the Hybrid Parametric Auditing method enables the reliable translation of modeling concepts expressed in natural language into consistent and geometrically sound parametric models. The approach demonstrated potential to optimize the early design stage, while overcoming geometric limitations inherent to generative artificial intelligence models, ensuring greater stability in the automated generation of BIM models.

Keywords: BIM. Parametric Modeling. Artificial Intelligence.

[1]Bacharel em Engenharia Civil pela Universidade Federal do Piauí (UFPI).
[2]Bacharel em Odontologia,Centro Universitário UNINOVAFAPI Afya.

RESUMEN: Este artículo tuvo como objetivo desarrollar un flujo de automatización para la modelación inicial en Building Information Modeling (BIM), integrando lenguaje natural y lenguaje de programación. La metodología consistió en un enfoque híbrido en el que un Modelo de Lenguaje Grande (LLM) actúa como intérprete semántico para estructurar datos en formato JSON, mientras que un algoritmo en IronPython opera como "auditor" y modelador a través de la API de Autodesk Revit. Este motor en Python aplica rigurosas validaciones geométricas, como cálculos de perímetro restringido, filtros de tolerancia de curvas y control de cotas de elevación. Los resultados demostraron que delegar el control topológico exclusivamente a la Inteligencia Artificial genera alucinaciones matemáticas y fallos estructurales, como la extrapolación de pilares que atraviesan los distintos niveles del edificio. No obstante, la intervención del código Python mitigó el 100% de estas inconsistencias, permitiendo la generación autónoma y precisa de esqueletos estructurales, aleros dinámicos y aberturas conceptuales. Se concluye que el método de Auditoría Paramétrica Híbrida permite la traducción confiable de conceptos de modelación expresados en lenguaje natural en modelos paramétricos consistentes y geométricamente íntegros. El enfoque demostró potencial para optimizar la fase de estudio preliminar, al mismo tiempo que supera las limitaciones geométricas inherentes a los modelos de inteligencia artificial generativa, garantizando mayor estabilidad en la generación automatizada de modelos BIM.

Palabras clave: BIM. Modelación Paramétrica. Inteligencia Artificial.

INTRODUÇÃO

A adoção do *Building Information Modeling* (BIM) revolucionou a Indústria da Arquitetura, Engenharia e Construção (AEC), substituindo a representação gráfica bidimensional por modelos tridimensionais paramétricos baseado em dados (EASTMAN CM, et al., 2018). Contudo, apesar do avanço tecnológico nas etapas de compatibilização e detalhamento, a fase de Estudo Preliminar e modelagem inicial ainda exige um alto esforço manual. A criação de malhas estruturais, níveis, paredes, lajes e outros elementos básicos exige operações repetitivas que dependem da interpretação humana de intenções de projeto frequentemente descritas em linguagem natural ou representadas por esquemas conceituais (VOLK R, et al., 2014).

Nos últimos anos, o estado da arte da automação na AEC tem convergido para a integração da Inteligência Artificial (IA), especificamente os *Large Language Models* (LLMs), como o ChatGPT da OpenAI. Pesquisas recentes demonstram que algoritmos de Processamento de Linguagem Natural (NLP) possuem elevada capacidade semântica para interpretar e extrair informações relevantes de documentos técnicos e especificações utilizadas na indústria da construção (DING L, et al., 2019). No entanto, a aplicação direta dessas ferramentas para a geração de geometria tridimensional encontra limitações estruturais graves, uma vez que os LLMs são, em sua essência, motores probabilísticos de predição de texto, com baixa consciência espacial ou raciocínio euclidiano.

O problema central desta pesquisa consiste na incompatibilidade entre a natureza estocástica e preditiva da IA generativa e a rigidez topológica exigida pelas plataformas BIM. Quando solicitados a traduzir comandos textuais diretamente para rotinas de modelagem paramétrica, os Grandes Modelos de Linguagem (LLMs) frequentemente apresentam déficits de raciocínio espacial e "alucinações matemáticas", resultando em acúmulo de erros geométricos (BORJI A, 2023). Estas falhas manifestam-se, por exemplo, na incapacidade de calcular fechamentos de perímetros exatos, na confusão de cotas de elevação no eixo Z e na geração de vetores de comprimento quase nulo. Tais imprecisões disparam erros sistêmicos de tolerância estrutural — como a exceção *ShortCurveTolerance* nativa da API do Revit. Consequentemente, a entrega do controle geométrico e topológico absoluto aos LLMs inviabiliza a geração de modelos semânticos que sejam funcionalmente utilizáveis na prática de projetos.

No contexto da modelagem paramétrica em plataformas BIM, estas inconsistências podem manifestar-se na geração de perímetros incorretos, vetores degenerados ou problemas na definição de níveis e elevações. Esses erros entram em conflito com as restrições geométricas impostas por *kernels* de modelagem presentes em softwares BIM, como as tolerâncias de curva mínima utilizadas pelo Autodesk Revit. Dessa forma, a delegação irrestrita da criação geométrica a modelos de linguagem compromete a integridade dos modelos gerados e limita sua aplicação prática no processo de projeto.

Diante desta lacuna, justifica-se a necessidade de um sistema de automação híbrido. Em vez de forçar a IA desenhar diretamente a geometria nativa, propõe-se a utilização de um intermediador em programação tradicional, capaz de receber a matriz de dados da IA e submetê-la a uma rigorosa validação antes da modelagem.

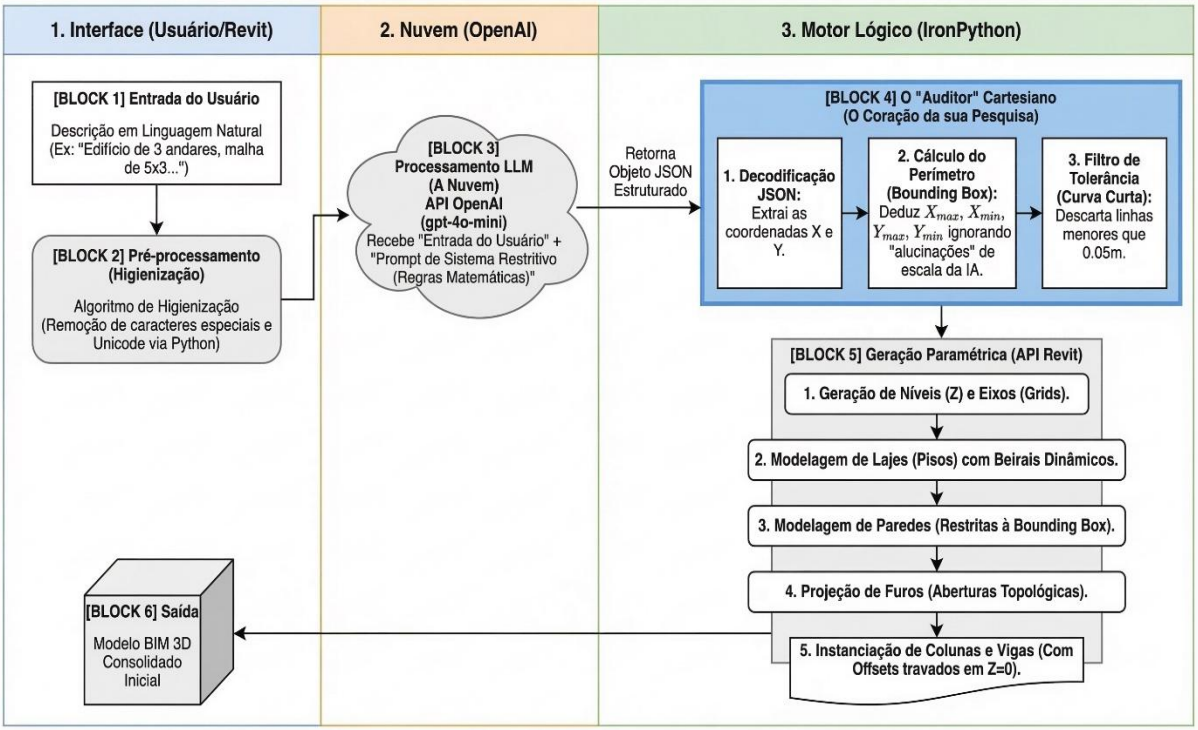
Neste contexto, o presente artigo tem como objetivo apresentar um estudo de caso focado no desenvolvimento de um *pipeline* automatizado para a etapa de modelagem inicial em BIM, integrando linguagem natural e linguagem de programação estruturada. A pesquisa busca demonstrar como a utilização de um LLM como extrator semântico (via objeto JSON), aliada a um *script* em *IronPython* atuando como "auditor" dentro do *software* Autodesk Revit, pode mitigar as alucinações da IA e gerar modelos paramétricos íntegros, otimizando o fluxo de trabalho arquitetônico de forma acessível e escalável.

MÉTODOLOGIA

A presente pesquisa caracteriza-se como um estudo de caso de natureza aplicada e exploratória, focado no desenvolvimento de um fluxo de trabalho automatizado para a etapa de concepção inicial da modelagem em plataformas BIM, como estruturado na Figura 1. O método foi estruturado na criação de um *script* capaz de traduzir instruções em linguagem natural para rotinas de geração de elementos construtivos.

Para avaliar o comportamento do sistema desenvolvido, foram realizados testes experimentais progressivos testando a complexidade das instruções em linguagem natural fornecidas ao modelo de linguagem. Os experimentos consistiram na geração automatizada de diferentes *layouts* geométricos em ambiente BIM, incluindo variações na quantidade de eixos estruturais, número de pavimentos e presença de elementos adicionais como beirais e aberturas. Ao todo, foram executados 15 testes experimentais variando a malha estrutural e parâmetros geométricos.

Figura 1: Fluxograma



Fonte: O autor (2026).

3.1. Ferramentas e fluxo base

O ecossistema desenvolvido baseia-se em três pilares tecnológicos fundamentais:

Interface e processamento (LLM): Utilizou-se a API da OpenAI, especificamente o modelo generativo *gpt-4o-mini*, configurado com temperatura 0.0 para maximizar o determinismo lógico e matemático das respostas, e minimizar a criatividade inerente ao modelo.

Ambiente de modelagem e informação (BIM): O *software* Autodesk Revit foi empregado como plataforma de tecnologia BIM, recebendo as instruções geométricas para a criação da modelagem base.

Ponte de Integração (*Middleware*): A biblioteca *pyRevit* foi utilizada para compilar o código escrito em *IronPython*, permitindo a integração direta entre os dados recebidos via *web* (API da OpenAI) e a API nativa do Revit (*Revit API*).

3.2. Engenharia de Prompt e Estruturação de Dados

Para contornar a imprevisibilidade típica dos Modelos de Linguagem Grande (LLMs) no tratamento de dados espaciais, desenvolveu-se um *System Prompt* restritivo. O principal desafio de integrar LLMs a *softwares* CAD/BIM é que modelos de linguagem são probabilísticos e geram texto discursivo, enquanto a modelagem geométrica exige vetores exatos e determinísticos. A IA foi instruída a não gerar texto discursivo, mas sim a atuar como um conversor de linguagem natural para uma estrutura de dados JSON (*JavaScript Object Notation*) predefinida.

O dicionário JSON foi projetado para conter chaves específicas, como: configuracoes (número de pavimentos, pé-direito), eixos (linhas de referência), beirais (extensões de laje por pavimento) e aberturas (furos nas paredes).

Para solucionar o problema clássico de erro de contagem (*Fencepost Error*) e impedir que a IA dependa demais dos exemplos citados (*Overfitting*), regras matemáticas rígidas foram embutidas no *prompt*. A IA foi treinada para calcular variáveis algébricas de uma malha ortogonal (N X M), onde N e M representam a quantidade exata de eixos verticais e horizontais, e não a quantidade de vãos, garantindo a extração precisa de coordenadas cartesianas (X, Y).

3.3. Tratamento de Dados

A comunicação entre o Revit e a API, feita por requisições HTTP (*WebClient*), exigiu um cuidado especial com a codificação dos textos. Para evitar erros, foi criada uma função chamada “limpar_texto”, que remove acentos, caracteres especiais e símbolos como "º" e "ª". Esse processo de sanitização foi necessário para a integração entre linguagem natural e de máquina, assim, evitando erros de leitura de texto (*UnicodeDecodeError*) durante o envio dos dados e garante o funcionamento estável da ferramenta.

3.4. Algoritmo de Geração BIM

A etapa final da metodologia consistiu no desenvolvimento do *script* em *IronPython* que executa a modelagem. Modelos de Linguagem Grande (LLMs) são, em sua essência, motores probabilísticos de texto e carecem de consciência espacial ou raciocínio euclidiano. Para evitar esse comportamento, o algoritmo não atua como um mero executor de comandos, mas foi programado estruturalmente para operar como um "auditor espacial". Antes de acionar a API nativa do Autodesk Revit, o *script* valida os dados do JSON e aplica quatro rigorosas regras de validação geométrica:

3.4.1. Geração do perímetro restrito

Durante os ensaios, observou-se que o LLM apresentava falhas de multiplicação ao dimensionar o fechamento do edifício (ex: ao solicitar 4 eixos espaçados a 4 metros, a IA inferia um tamanho total de 16 metros, ignorando que 4 eixos geram apenas 3 vãos, totalizando 12 metros). Para evitar que paredes e lajes fossem geradas sem a referência correta, além da área estrutural planejada, o algoritmo invalida proativamente qualquer cálculo de perímetro feito pela IA. O código varre exclusivamente as coordenadas matemáticas da malha de eixos, extrai os valores extremos ($X_{min}, X_{max}, Y_{min}, Y_{max}$) e constrói uma "Caixa Cartesiana Sanitizada". É estritamente sobre os limites desse polígono perfeito que as paredes de fechamento, lajes, vigas e pilares são forçadas a ser instanciadas, garantindo que a modelagem envolva perfeitamente o esqueleto estrutural.

3.4.2. Tolerância de curvas curtas (*Short Curve Tolerance*)

A formulação geométrica do Revit possui uma trava de segurança intrínseca que impede a modelagem de elementos com dimensões microscópicas (inferiores a aproximadamente 0,8 milímetros), disparando um problema de exceção (*ArgumentsInconsistentException*) que aborta a rotina. Devido a alucinações matemáticas da IA ao calcular cruzamentos, o JSON ocasionalmente fornecia linhas com os pontos de início e fim idênticos (comprimento igual a zero). O algoritmo contorna essa limitação aplicando um cálculo de distância euclidiana (*DistanceTo*) antes de qualquer criação. Vetores com comprimento inferior a 0.05 metros são automaticamente filtrados e descartados do processamento, prevenindo o travamento do sistema.

3.4.3. Eixo Z e níveis

A criação de pilares e vigas (*Family Instances*) via API exigem duas determinações: a indicação de uma coordenada espacial tridimensional (X, Y, Z) e a vinculação a um nível do modelo (*Level*). Identificou-se que criar elementos informando a coordenada $Z = 0$ combinado com um nível superior (ex: Nível 2, elevação 3.0m) levava o Revit a deduzir erroneamente um deslocamento de base negativo, fazendo com que pilares transpassassem outros pavimentos de forma contínua. Para instituir a "disciplina de alturas", o algoritmo toma a elevação absoluta do nível atual (*Level.Elevation*) e cria o elemento na coordenada Z correta. Simultaneamente, o código acessa os parâmetros internos do elemento

(*BuiltInParameter.FAMILY_BASE_LEVEL_OFFSET_PARAM*) e o determina para o valor 0.0, garantindo a segmentação correta da estrutura pavimento a pavimento.

3.4.4. Projeção vetorial para aberturas

Para a inserção de vãos conceituais de portas ou janelas, optou-se por não depender do carregamento de famílias complexas (que exigiriam *templates* específicos). Dessa forma, foi adotado o método nativo *NewOpening* da API. Como a IA fornece apenas uma coordenada 2D aproximada para o furo, o algoritmo calcula a distância da coordenada fornecida em relação a todas as paredes do pavimento, identifica a face mais próxima, e projeta o centro do furo exatamente sobre o vetor da parede. A partir desse centro, o algoritmo deduz as extremidades

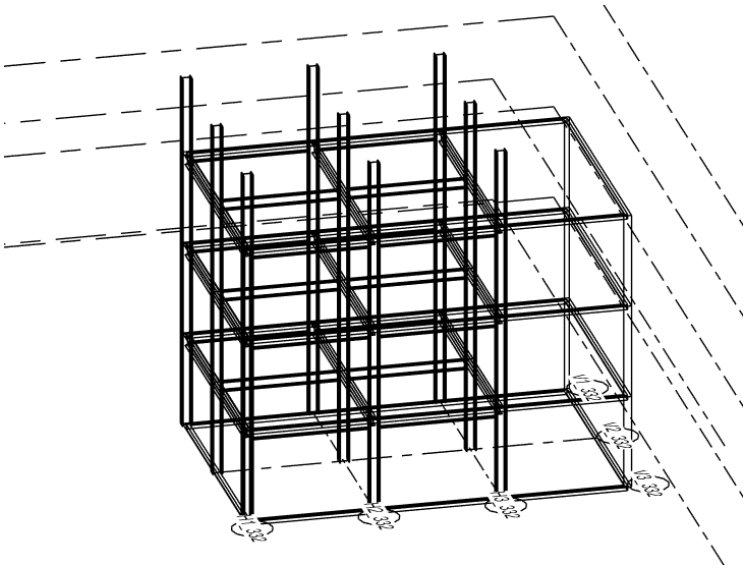
inferior esquerda e superior direita com base na largura, altura e peitoril informados pelo usuário, ordenando que o Revit realize a abertura diretamente na geometria da parede.

RESULTADOS

Para validar a eficácia do fluxo de trabalho desenvolvido, foram realizados testes progressivos de modelagem via linguagem natural. Os testes focaram em avaliar tanto a capacidade semântica do LLM em interpretar as intenções de projeto, quanto a eficácia do motor de auditoria em *Python* ao lidar com as inconsistências espaciais geradas pela IA.

Nos testes iniciais, foi solicitado à IA a geração de uma malha ortogonal simples e o respectivo fechamento do perímetro (por exemplo: *"Malha de 4x4 eixos espaçados a 4m"*). Observou-se que a LLM compreende a instrução textual, mas falha criticamente ao converter essa instrução para a representação geométrica no plano euclidiano, manifestando o clássico Erro do Poste e Cerca (*Fencepost Error*). A IA gerou as coordenadas dos eixos estruturais corretamente (0m, 4m, 8m e 12m). Contudo, ao calcular o perímetro para as paredes de fechamento, o modelo assumiu uma extrapolação algébrica simplória ($4x4 = 16$ metros), estendendo as paredes 4 metros além do último eixo estrutural (Figura 2).

Figura 2: Geração inicial demonstrando o erro do poste e cerca devido à alucinação matemática do LLM

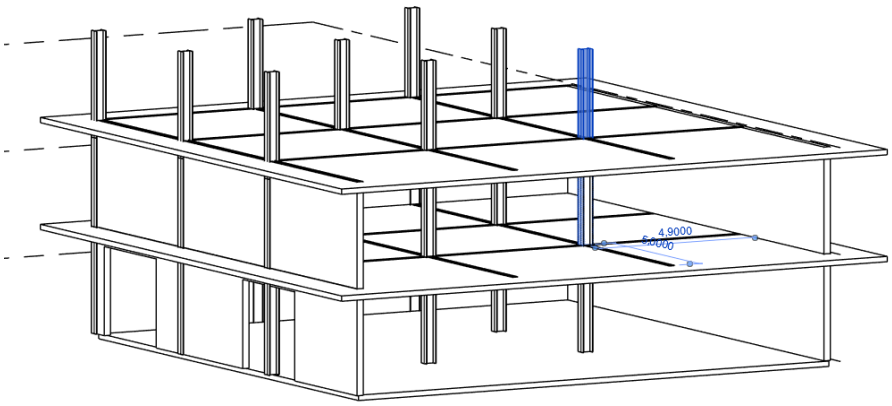


Fonte: O autor (2026).

A implementação da "Caixa Cartesiana Sanitizada" na camada Python provou-se eficaz, suprimindo integralmente essa extrapolação nas iterações seguintes e forçando o fechamento perfeito da modelagem arquitetônica.

A integração direta entre coordenadas geradas por IA e a API do Revit revelou comportamentos atípicos na criação de pilares e vigas. Quando a IA instruía a inserção de pilares em níveis superiores (ex: Andar 1), mas fixava a coordenada em $Z = 0$, a API do Revit compensava essa divergência aplicando um deslocamento de base (*Base Offset*) negativo equivalente à altura do pavimento (Figura 3).

Figura 3: O "Efeito Z=0" causando o transpasse de pilares entre pavimentos devido à ausência de restrição de elevação.



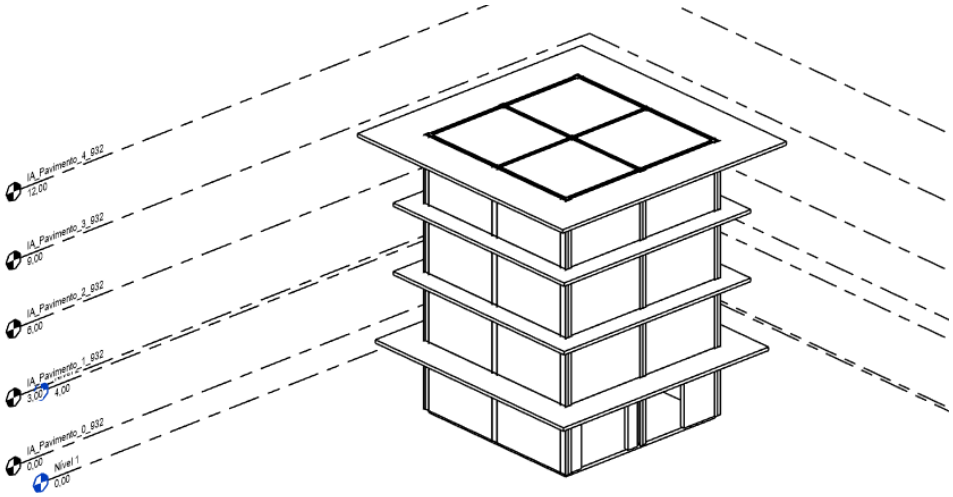
Fonte: O autor (2026).

Para correção, foi necessário transformar o motor de processamento do *Python* para atuar como mediador de cotas. Ao forçar a leitura do parâmetro *Level.Elevation* e zerar os *offsets* via código, a estrutura passou a ser segmentada corretamente. Além disso, o filtro de tolerância geométrica criado no código eliminou com sucesso as exceções críticas de "Curva Curta" (*ShortCurveTolerance*) que corrompiam o *script* quando a IA alucinava vetores de comprimento nulo.

O teste de estresse avaliou a capacidade do sistema em lidar com variações morfológicas complexas simultâneas, processando o seguinte *prompt*: *"Crie um prédio de 4 andares com malha de 5x3 eixos espaçados a 4m. Quero uma marquise de 1.5 metros na laje do 1º andar, uma marquise de 1.0 metro na laje do 2º andar, uma marquise de 1.0 metro na laje do 3º andar e um beiral grande de 2.0 metros na cobertura. Adicione 2 aberturas grandes no térreo"*. O sistema obteve sucesso integral. O algoritmo em *Python* percorreu a matriz em formato JSON e aplicou, de forma independente,

as expansões de perímetro para cada laje. Como resultado, foi gerada a volumetria correspondente, conforme ilustrado na Figura 4.

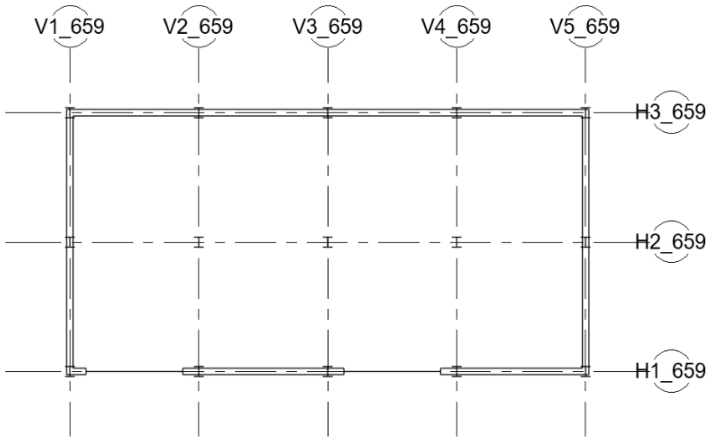
Figura 4: Modelo paramétrico 3D final com beirais escalonados independentes



Fonte: O autor (2026).

As aberturas conceituais foram projetadas topologicamente nas faces corretas, validando o uso do método nativo *NewOpening* utilizando cálculos vetoriais em Python, dispensando famílias pré-carregadas. Em vista de representação técnica, as linhas de eixo (*Grids*) foram geradas cruzando a totalidade do edifício, com o deslocamento estético (*offset*) calculado matematicamente para posicionar as identificações fora da projeção da arquitetura, entregando uma planta baixa limpa (Figura 5).

Figura 5: Planta baixa estrutural com eixos estendidos automaticamente e bolhas de identificação recuadas.



Fonte: O autor (2026).

Os resultados atestam que a tentativa de transferir o controle geométrico diretamente ao LLM resulta em modelos corrompidos que violam as regras da construção civil. A abordagem de "Auditoria" desenvolvida, onde a IA atua apenas como provedor lógico matricial e o Python assume a responsabilidade da geometria descritiva, resulta em um método viável e escalável. Como limitação, o estudo focou na fase de Estudo Preliminar, indicando como caminhos para novas pesquisas a expansão do código para especificações quantitativas e detalhamento de materiais construtivos integrados ao LLM.

CONCLUSÃO

O presente estudo demonstrou a viabilidade técnica e a eficácia da integração entre Modelos de Linguagem Grande (LLMs) e a programação em Python para a automação da modelagem inicial em plataformas BIM. A pesquisa evidenciou que a interpretação de intenções de projeto descritas em linguagem natural para modelos BIM tridimensionais não apenas é possível, como apresenta um imenso potencial para otimizar a fase de Estudo Preliminar na arquitetura e engenharia.

Os testes demonstraram que a delegação irrestrita do controle topológico a um motor probabilístico (como o *GPT-4o-mini*) pode resultar em erros matemáticos críticos. Entre os problemas observados estão a extrapolação indevida de perímetros e a perda de consistência na dimensão de elevação.

Diante disso, a grande contribuição deste trabalho é a consolidação do método de Auditoria Paramétrica Híbrida. Nesse modelo, o LLM passa a atuar como extrator semântico e estruturador de dados em formato JSON, enquanto o algoritmo em *IronPython* atua como auditor cartesiano, o sistema conseguiu mitigar 100% das inconsistências espaciais. A aplicação de regras rígidas de restrição de perímetro, tolerância de curvas e nivelamento forçado garantiu a geração de um modelo BIM íntegro e perfeitamente alinhado às diretrizes da construção civil.

Do ponto de vista prático, a ferramenta desenvolvida democratiza o acesso à modelagem paramétrica complexa, permitindo que profissionais gerem esqueletos estruturais, lajes com beirais dinâmicos conceituais em questão de segundos, utilizando apenas comandos textuais simples.

Como sugestões para trabalhos futuros, propõe-se o desenvolvimento de mecanismos mais avançados de geração de aberturas arquitetônicas baseadas em análises climáticas, incorporando parâmetros como orientação solar, ventilação predominante e desempenho

térmico. Recomenda-se também a implementação de algoritmos capazes de gerar layouts espaciais mais complexos a partir de mapas de bolhas, permitindo a tradução automática de diagramas conceituais em arranjos arquitetônicos preliminares. Por fim, destaca-se o potencial de criação de módulos bidirecionais, nos quais a IA não apenas realize a modelagem, mas também seja capaz de interrogar o modelo BIM e extrair quantitativos e informações construtivas por meio de linguagem natural.

REFERÊNCIAS

BORJI A. A categorical archive of ChatGPT failures. *arXiv*, 2023.

DING L, ZHONG B, WU S, et al. Construction risk knowledge management in BIM using ontology and natural language processing. *Automation in Construction*, 2019; 99: 14-28.

EASTMAN CM, TEICHOLZ P, SACKS R, et al. BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers. 3rd ed. Hoboken: John Wiley & Sons, 2018; 688p.

VOLK R, et al. Building Information Modeling (BIM) for existing buildings — Literature review and future needs. *Automation in Construction*, 2014; 38: 109-127.

YU Y, et al. Evaluating ChatGPT on Korea's BIM Expertise Exam and improving its performance through RAG. *Journal of Computational Design and Engineering*, 2025; 12(4): 94-120.