XXI ENTAC – Encontro Nacional de Tecnologia do Ambiente Construído
14 a 16 de outubro de 2026 | Goiânia-Brasil
DOI: https://doi.org/10.46421/entac.v21i1.8522

Verificação de modelos utilizando o padrão information delivery specification (IDS)

Model verification using the information delivery specification (IDS) standard

Matheus Brondani Prevedello
Universidade Federal de Santa Maria | Santa Maria| Brasil | matheusfx95@gmail.com
Maria Eduarda Bueno Silva
Universidade Federal de Santa Maria | Santa Maria| Brasil | madubuenosilva@gmail.com
Vitor de Oliveira Benvegnu
Universidade Federal de Santa Maria | Santa Maria | Brasil |vitorbenvegnu17@gmail.com
João Vitor Lichtenecker de Almeida
Universidade Federal de Santa Maria | Santa Maria | Brasil | joao.lichtenecker@acad.ufsm.br
Eduardo Cesar Pachla
Universidade Federal de Santa Maria | Santa Maria - RS | Brasil | eduardo.pachla@ufsm.br

Resumo

Este artigo investiga a aplicação do padrão Information Delivery Specification (IDS) como ferramenta para a automação da auditoria de modelos digitais. A metodologia consistiu na tradução de requisitos de informação para o formato XML e na execução da auditoria por meio de algoritmos customizados desenvolvidos em linguagem Python com a biblioteca IfcOpenShell. Os resultados demonstram que o IDS atua como um filtro sintático eficaz no ecossistema BIM, permitindo a detecção imediata de omissões de informações. A integração via programação possibilitou a automação do ciclo de coordenação através da geração de relatórios BCF com geolocalização de falhas. Conclui-se que o uso do IDS, seja em softwares consolidados ou em soluções programadas, assegura a integridade dos dados, promove a soberania tecnológica e oferece uma alternativa escalável para a gestão da qualidade de modelos openBIM.

Palavras-chave: BIM. IDS. Automação da Verificação. Conformidade de Requisitos.

Abstract

This article investigates the application of the Information Delivery Specification (IDS) standard as a tool for automating the auditing of digital models. The methodology consisted of translating information requirements into XML format and performing the audit through customized algorithms developed in Python using the IfcOpenShell library. The results demonstrate that IDS acts as an effective syntactic filter within the BIM ecosystem, enabling the immediate detection of Fire Rating omissions. Integration through programming enabled the automation of the coordination cycle by generating BCF reports with geolocated issues. It is concluded that the use of IDS ensures data integrity and provides a scalable alternative for quality management in openBIM models.

Keywords: BIM. IDS. Automation. IfcOpenShell. BIMcollab.

PREVEDELLO, M. B.; SILVA, M. E. B.; BENVEGNU, V. O.; ALMEIDA, J. V. L.; PACHLA, E. C. Verificação de modelos utilizando o padrão Information Delivery Specification (IDS). In: ENCONTRO NACIONAL DE TECNOLOGIA DO AMBIENTE CONSTRUÍDO, 21, 2026, Goiânia. **Anais**... Goiânia: ANTAC, 2026.

INTRODUÇÃO

A crescente digitalização da indústria da Arquitetura, Engenharia e Construção (AEC) tem ampliado o uso do *Building Information Modeling* (BIM) como meio de garantir a qualidade e a conformidade das informações ao longo do ciclo de vida dos projetos. Nesse cenário, um dos principais desafios consiste em assegurar que os requisitos de informação sejam corretamente representados nos modelos digitais, uma vez que ainda são frequentemente definidos em formatos não automatizáveis, como documentos textuais e planilhas, exigindo replicação manual e resultando em baixa produtividade, fragmentação do processo projetual e retrabalho, com impactos diretos em custos, prazos e qualidade das edificações [1].

No Brasil, a adoção do BIM foi institucionalizada por meio do Decreto nº 10.306, que estabelece diretrizes para sua implementação em obras e serviços de engenharia da administração pública federal. Essa regulamentação implica a produção de modelos com elevado volume de informações, o que torna inviável que profissionais realizem manualmente a validação completa dos dados, especialmente em empreendimentos de maior porte e complexidade, devido à suscetibilidade a erros humanos, ao elevado custo de tempo, à possibilidade de geração de atrasos, à falta de padronização na comunicação e à consequente redução da produtividade [2].

Diante desse cenário, a automação dos processos de verificação torna-se essencial para garantir a conformidade dos modelos com os requisitos estabelecidos, reduzindo a dependência de verificações manuais e minimizando a ocorrência de erros. Além de otimizar o tempo e os recursos envolvidos, a automatização contribui para maior padronização das informações e para o aumento da confiabilidade dos dados utilizados nas etapas de projeto, execução e operação [3].

Sob esse olhar, este artigo tem como objetivo investigar como requisitos de informação podem ser verificados automaticamente em modelos BIM por meio do padrão *Information Delivery Specification* (IDS), explorando sua aplicação como instrumento de estruturação e validação de dados. Para isso, propõe-se a utilização de um código desenvolvido em Python integrado ao IDS, permitindo a verificação automatizada dos requisitos preestabelecidos e contribuindo para a melhoria da qualidade informacional dos modelos digitais.

REVISÃO BIBLIOGRÁFICA

A fundamentação teórica deste estudo baseia-se na premissa de que a eficiência do BIM depende da transição da modelagem geométrica para a gestão da informação estruturada e interoperável [4]. Nesse contexto, a literatura aponta que a fragmentação do setor de Arquitetura, Engenharia e Construção (AEC) exige métodos capazes de garantir a integridade dos dados ao longo do ciclo de vida do ativo, transformando requisitos abstratos em entregas digitais verificáveis[5][6].

GESTÃO DA INFORMAÇÃO NO BIM

A gestão da informação em BIM, conforme a NBR ISO 19650, configura-se como um fluxo ao longo do ciclo de vida da edificação, organizando a produção e verificação de dados em escalas e garantindo à tomada de decisão por meio do Ambiente Comum de Dados (CDE) [6]. Nesse sistema, os *Organizational Information Requirements* (OIR) definem necessidades, enquanto os *Asset Information Requirements* (AIR) especificam os dados necessários para operação e manutenção do ativo [7].

No âmbito do projeto, os *Project Information Requirements* (PIR) detalham as informações necessárias para atender aos objetivos estratégicos, e os *Exchange Information Requirements* (EIR) estabelecem requisitos de produção, entrega e validação dos dados [8]. O Plano de Execução BIM (BEP) operacionaliza diretrizes, definindo responsabilidades e padrões de modelagem, garantindo consistência e controle de qualidade [9].

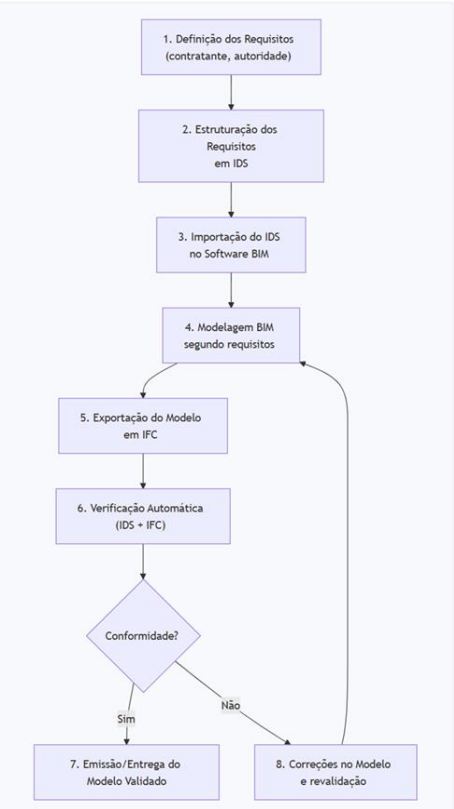
Para viabilizar a verificação desses requisitos, o *Information Delivery Specification* (IDS) atua como um formato aberto e interpretável por computador, permitindo a automação da conformidade por meio da definição de propriedades e classificações. Avanços, como filtros, ampliam sua aplicação para verificações complexas, incluindo códigos normativos [10].

GOVERNANÇA DE DADOS: RESPONSABILIDADES E FLUXOS DE TRABALHO

A governança de dados no BIM fundamenta-se na hierarquia de requisitos ISO 19650 (OIR → AIR → EIR), na qual a parte requerente (*Appointing Party*) define as necessidades e critérios de aceitação, formalizados no EIR [11]. Nesse contexto, o IDS atua como instrumento de governança ao materializar essa relação, sendo a parte requerente responsável pela definição dos requisitos e a parte fornecedora (*Appointed Party*) pela produção de informações em conformidade com essas exigências.

Dessa forma, o IDS estabelece uma conexão entre a definição da necessidade e a entrega, garantindo a transparência do fluxo informacional, a conformidade do ativo digital e a redução de conflitos quanto à qualidade dos dados. A operacionalização desse processo, desde a definição dos requisitos até a validação em ambiente openBIM, é sintetizada no fluxo de trabalho apresentado na Figura 1.

Figura 1: Fluxo de trabalho, elaborado com auxílio de inteligência artifical via LeapSpace, para verificação de conformidade de modelos BIM baseada no padrão IDS.



Fonte: autores.

FORMATO IFC (INDUSTRY FOUNDATION CLASSES)

O IFC é um padrão internacional aberto e orientado a objetos, reconhecido pela ISO 16739-1:2018, que estrutura a informação por meio de entidades, atributos e relacionamentos complexos, permitindo representar tanto a geometria quanto as relações semânticas dos elementos [12]. A informação é organizada entre atributos intrínsecos e propriedades agrupadas em *Property Sets*, conectadas por relações como *IfcRelDefinesByProperties*, garantindo rastreabilidade e consistência dos dados [13][14].

Apesar disso, sua aplicação prática enfrenta desafios decorrentes da complexidade do esquema e da heterogeneidade dos dados, o que pode gerar interpretações ambíguas e lacunas informacionais. Dessa forma, o uso do IFC isoladamente não garante a qualidade dos dados, sendo necessária a adoção de mecanismos adicionais de validação e filtragem [15].

INFORMATION DELIVERY SPECIFICATION (IDS)

Como resposta às ambiguidades do esquema IFC, surge o padrão *Information Delivery Specification* (IDS) é o padrão aberto da *buildingSMART International* projetado para a formalização de requisitos de informação em modelos digitais. Estruturado em XML, o IDS permite a definição explícita e legível por máquina (*computer-interpretable*) de

requisitos associados ao esquema IFC, viabilizando a verificação automática de conformidade de forma precisa e acurada [5].

Diferente dos métodos baseados em planilhas manuais, o IDS utiliza um esquema semântico estruturado em especificações (*specifications*) para requisitos alfanuméricos. Sua arquitetura organiza-se em duas seções: a *applicability* (aplicabilidade), que atua como um filtro para identificar as entidades do modelo às quais a regra se aplica, e os *requirements* (requisitos), que descrevem as informações a serem verificadas nos elementos filtrados[16].

Essa estrutura opera por meio de facetas (*facets*), que permitem filtrar e validar elementos com base em sua entidade IFC, classificação, propriedades, materiais ou localização. O uso dessas facetas garante que a regra de intercâmbio seja aplicada estritamente aos objetos pretendidos, assegurando a precisão da auditoria automatizada[16].

Em síntese, o padrão funciona como o elo vital entre a definição estratégica e a validação operacional. Ao estruturar os dados de forma verificável, o IDS sustenta práticas de gestão da informação robustas, garantindo a integridade e a qualidade das entregas no setor da construção [16].

METODOLOGIA

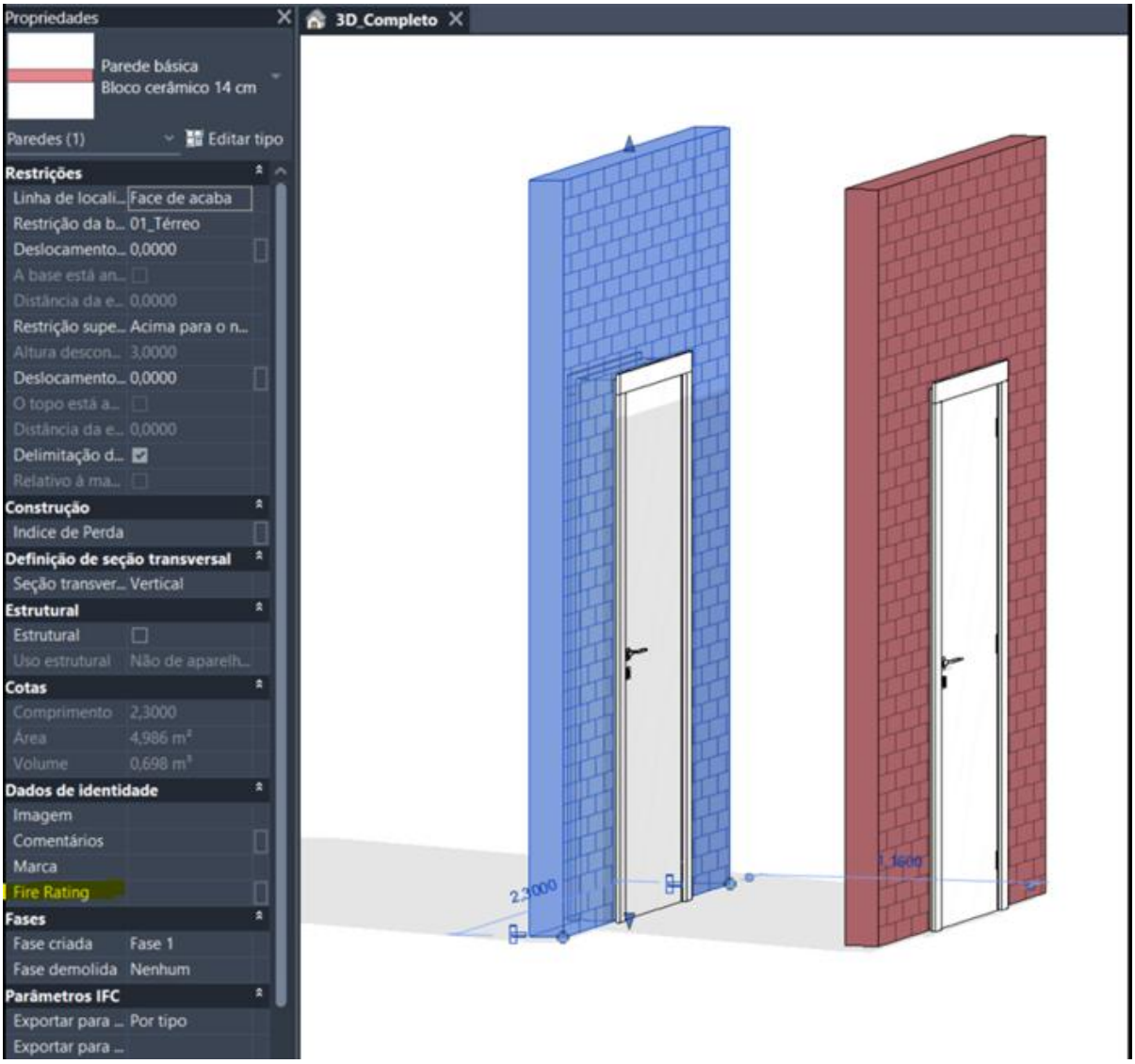
Este estudo caracteriza-se como uma pesquisa aplicada de natureza descritiva, cujo objetivo foi desenvolver um meio de verificação automatizada baseada na utilização do IDS e ferramentas de código aberto, a fim de validar a eficácia do padrão IDS na garantia da integridade e qualidade da informação.

PROCEDIMENTOS METODOLÓGICOS

A validação automatizada foi conduzida a partir de um fluxo estruturado em quatro etapas principais:

Etapa 1 – Modelagem e definição de atributos: foi realizada a modelagem de um elemento construtivo: paredes modeladas no software Autodesk Revit , como exposto na Figura 2. Posteriormente foi criada a propriedade *Fire Rating* (Resistência ao Fogo) como um parâmetro de instância, já que o desempenho de resistência ao fogo pode variar conforme a compartimentação e a localização específica do elemento no edifício, independentemente da sua tipologia ou família.

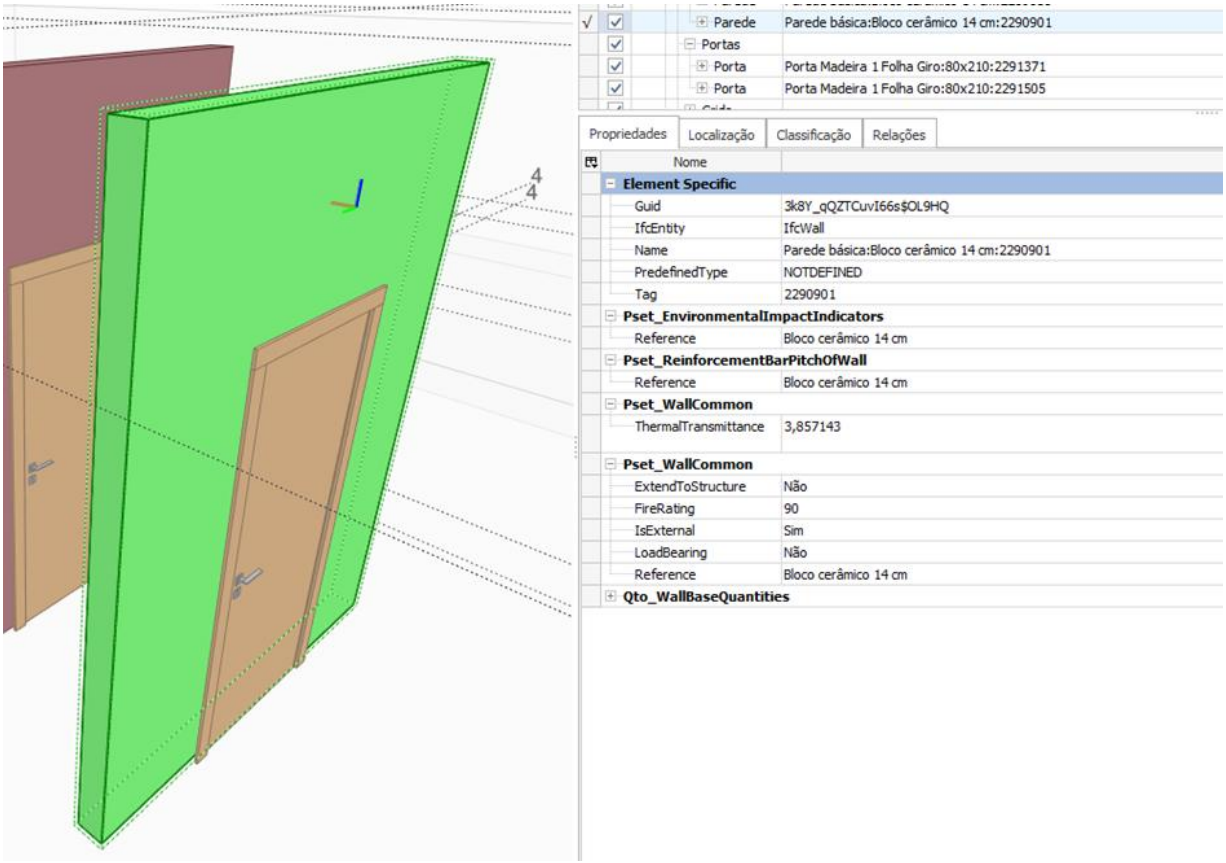
Figura 2: Modelagem no Revit e configuração de parâmetros de instância.



Fonte: autores.

Etapa 2 – Exportação para o IFC 4: a transição do ambiente de modelagem proprietário para o ecossistema openBIM foi operacionalizada por meio da exportação do modelo para o esquema de dados IFC 4, o que garante que a granularidade da estrutura execute a auditoria do protocolo IDS (*Information Delivery Specification*) com precisão. Esta etapa foi determinante para assegurar a persistência da informação durante a migração entre plataformas. Além disso, para garantir a semântica do mapeamento de dados, as propriedades foram padronizadas pela *buildingSMART*: o *Pset_WallCommon* para as paredes do modelo, o que pode ser observado na Figura 3.

Figura 3: Atributos de instância exportados no esquema IFC para a entidade ifcWall – Visualização via software BIM Vision.



Fonte: autores.

Etapa 3 – Estruturação da *Information Delivery Specification* (IDS):

O arquivo IDS foi inicialmente estruturado utilizando o software BIMcollab Zoom, empregado como interface gráfica para geração da estrutura básica do documento XML. A escolha dessa ferramenta deveu-se à facilidade de configuração das especificações IDS e à exportação direta do arquivo em conformidade com o esquema definido pela buildingSMART. Embora o ecossistema BlenderBIM disponibilize o módulo *IfcTester* para criação e validação de especificações IDS, optou-se por não utilizá-lo diretamente, pois o objetivo deste estudo não consistiu na avaliação comparativa de interfaces de autoria, mas no desenvolvimento de um fluxo de auditoria automatizada por programação. Dessa forma, o BIMcollab foi utilizado para acelerar a estruturação inicial do IDS, enquanto a interpretação, validação e processamento das regras foram executados em ambiente Python, utilizando o módulo *IfcTester* (*IfcOpenShell*).

Na especificação técnica, definiu-se a entidade *IfcWall* como alvo no esquema IFC, com obrigatoriedade do conjunto *Pset_WallCommon*. Como lógica de verificação, aplicou-se o operador “is defined” à propriedade *FireRating* (Figura 4), permitindo ao

algoritmo identificar elementos com campos vazios e classificá-los como não conformes.

Figura 4: Configuração da especificação IDS para paredes (Fire Rating Wall) no BIMcollab



Fonte: autores.

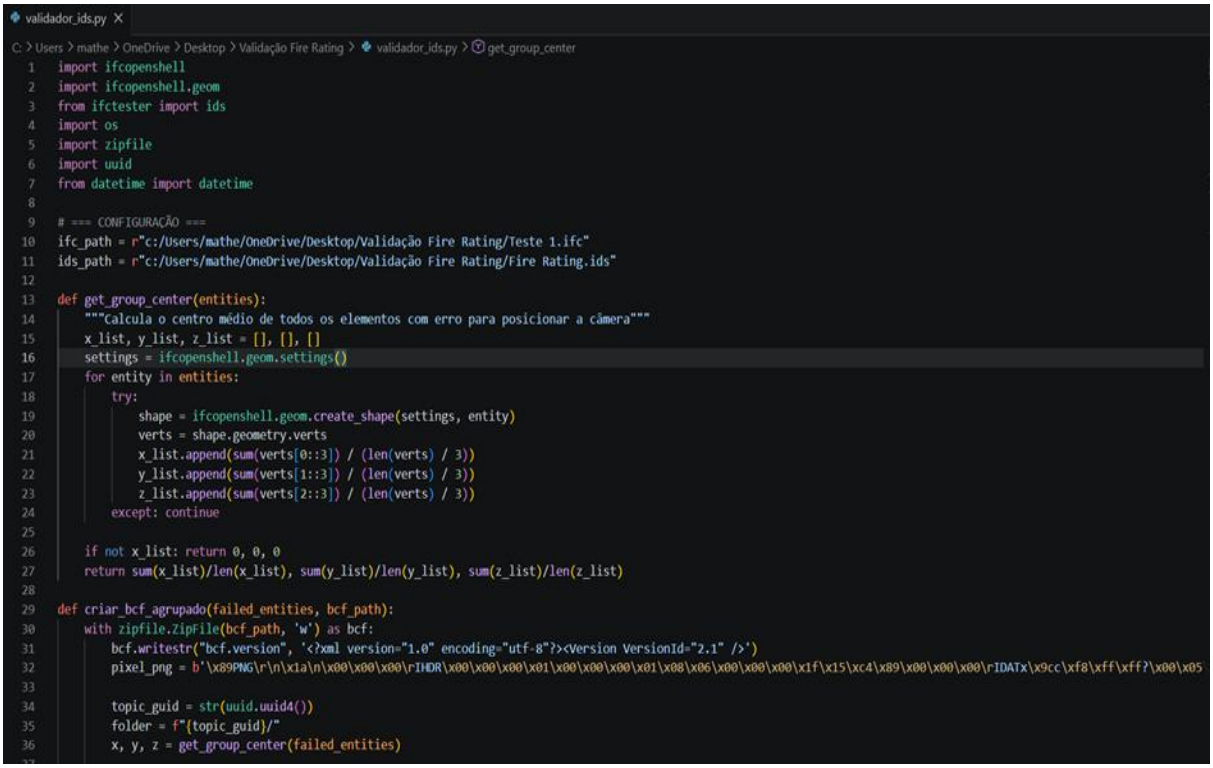
Etapa 4 – Validação Automatizada via Python e Ecossistema IfcOpenShell:

Para a implementação da via de auditoria baseada em ferramentas de código aberto, utilizou-se o ambiente de programação Python 3.12 integrado ao ecossistema *IfcOpenShell*, escolhido por sua robustez na manipulação do esquema IFC, permitindo acesso direto aos dados do modelo sem interfaces proprietárias. O desenvolvimento e a depuração dos scripts foram realizados no editor de código Visual Studio Code (VS Code) utilizando extensões específicas para Python, com auxílio de inteligência artifical via Google Gemini para manipulação dos códigos, que permitiram o monitoramento de variáveis e a execução de testes em tempo real. O fluxograma é apresentado na Figura 5.

Ressalta-se que o módulo IfcTester empregado neste estudo corresponde ao mesmo núcleo de validação disponibilizado no ecossistema BlenderBIM, sendo utilizado diretamente via API Python para possibilitar sua integração com rotinas personalizadas de geração de relatórios HTML e arquivos BCF (Figura 6).

A utilização do VS Code como ambiente de desenvolvimento facilitou a integração do código com bibliotecas de automação. O módulo *Ifctester* foi utilizado para interpretar a especificação IDS (XML) e compará-la com o modelo IFC, onde o script realizou a leitura simultânea do modelo digital e do arquivo de requisitos.

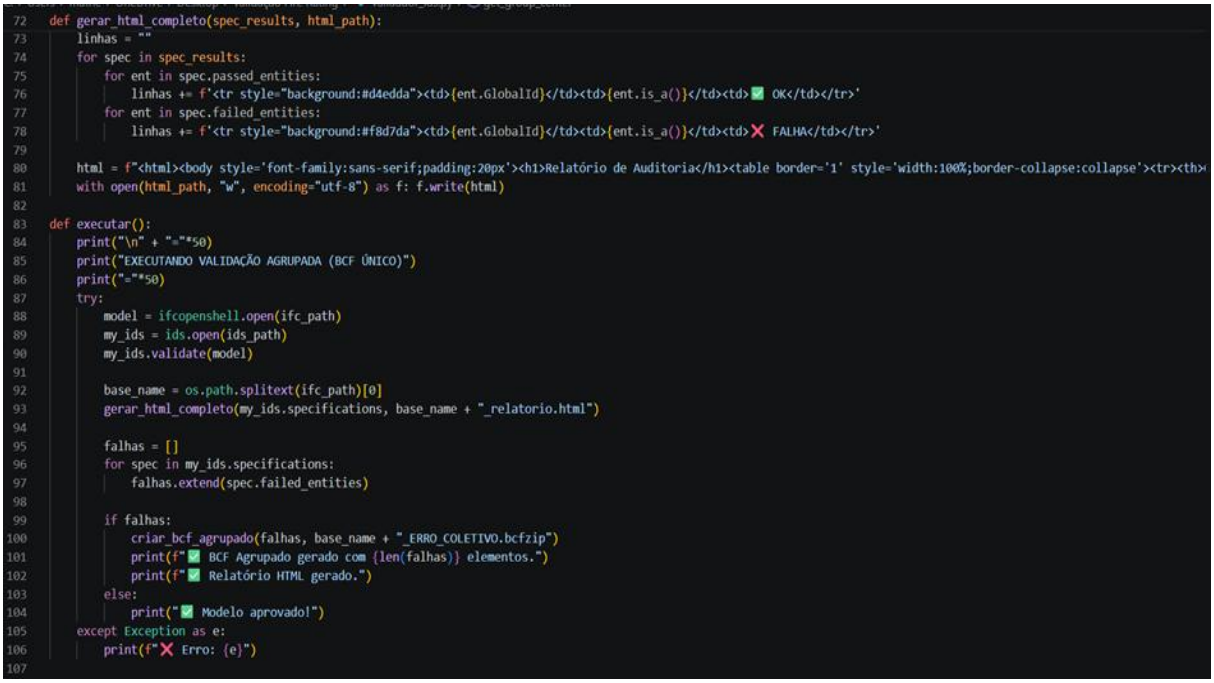
Figura 5: Fluxograma do processo de automação, detalhando a entrada dos arquivos IFC e IDS



```python
import ifcopenshell
import ifcopenshell.geom
from ifctester import ids
import os
import zipfile
import uuid
from datetime import datetime

# === CONFIGURAÇÃO ===
ifc_path = r"c:/Users/mathe/OneDrive/Desktop/Validação Fire Rating/Teste 1.ifc"
ids_path = r"c:/Users/mathe/OneDrive/Desktop/Validação Fire Rating/Fire Rating.ids"

def get_group_center(entities):
    """Calcula o centro médio de todos os elementos com erro para posicionar a câmera"""
    x_list, y_list, z_list = [], [], []
    settings = ifcopenshell.geom.settings()
    for entity in entities:
        try:
            shape = ifcopenshell.geom.create_shape(settings, entity)
            verts = shape.geometry.verts
            x_list.append(sum(verts[0::3]) / (len(verts) / 3))
            y_list.append(sum(verts[1::3]) / (len(verts) / 3))
            z_list.append(sum(verts[2::3]) / (len(verts) / 3))
        except: continue

    if not x_list: return 0, 0, 0
    return sum(x_list)/len(x_list), sum(y_list)/len(y_list), sum(z_list)/len(z_list)

def criar_bcf_agrupado(failed_entities, bcf_path):
    with zipfile.ZipFile(bcf_path, 'w') as bcf:
        bcf.writestr("bcf.version", '<?xml version="1.0" encoding="utf-8"?><Version VersionId="2.1" />')
        pixel_png = b'\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x00\x01\x00\x00\x00\x01\x08\x06\x00\x00\x00\x1f\x15\xc4\x89\x00\x00\x00\rIDATx\x9cc\xf8\xff\xff?\x00\x05

        topic_guid = str(uuid.uuid4())
        folder = f"{topic_guid}/"
        x, y, z = get_group_center(failed_entities)
```

Fonte: autores.

Figura 6 - Processamento lógico em Python e a geração das saídas em HTML e BCF



```python
def gerar_html_completo(spec_results, html_path):
    linhas = ""
    for spec in spec_results:
        for ent in spec.passed_entities:
            linhas += f'<tr style="background:#d4edda"><td>{ent.GlobalId}</td><td>{ent.is_a()}</td><td>✅ OK</td></tr>'
        for ent in spec.failed_entities:
            linhas += f'<tr style="background:#f8d7da"><td>{ent.GlobalId}</td><td>{ent.is_a()}</td><td>❌ FALHA</td></tr>'

    html = f"<html><body style='font-family:sans-serif;padding:20px'><h1>Relatório de Auditoria</h1><table border='1' style='width:100%;border-collapse:collapse'><tr><th>
    with open(html_path, "w", encoding="utf-8") as f: f.write(html)

def executar():
    print("\n" + "="*50)
    print("EXECUTANDO VALIDAÇÃO AGRUPADA (BCF ÚNICO)")
    print("="*50)
    try:
        model = ifcopenshell.open(ifc_path)
        my_ids = ids.open(ids_path)
        my_ids.validate(model)

        base_name = os.path.splitext(ifc_path)[0]
        gerar_html_completo(my_ids.specifications, base_name + "_relatorio.html")

        falhas = []
        for spec in my_ids.specifications:
            falhas.extend(spec.failed_entities)

        if falhas:
            criar_bcf_agrupado(falhas, base_name + "_ERRO_COLETIVO.bcfzip")
            print(f"✅ BCF Agrupado gerado com {len(falhas)} elementos.")
            print(f"✅ Relatório HTML gerado.")
        else:
            print("✅ Modelo aprovado!")
    except Exception as e:
        print(f"❌ Erro: {e}")
```

Fonte: autores.

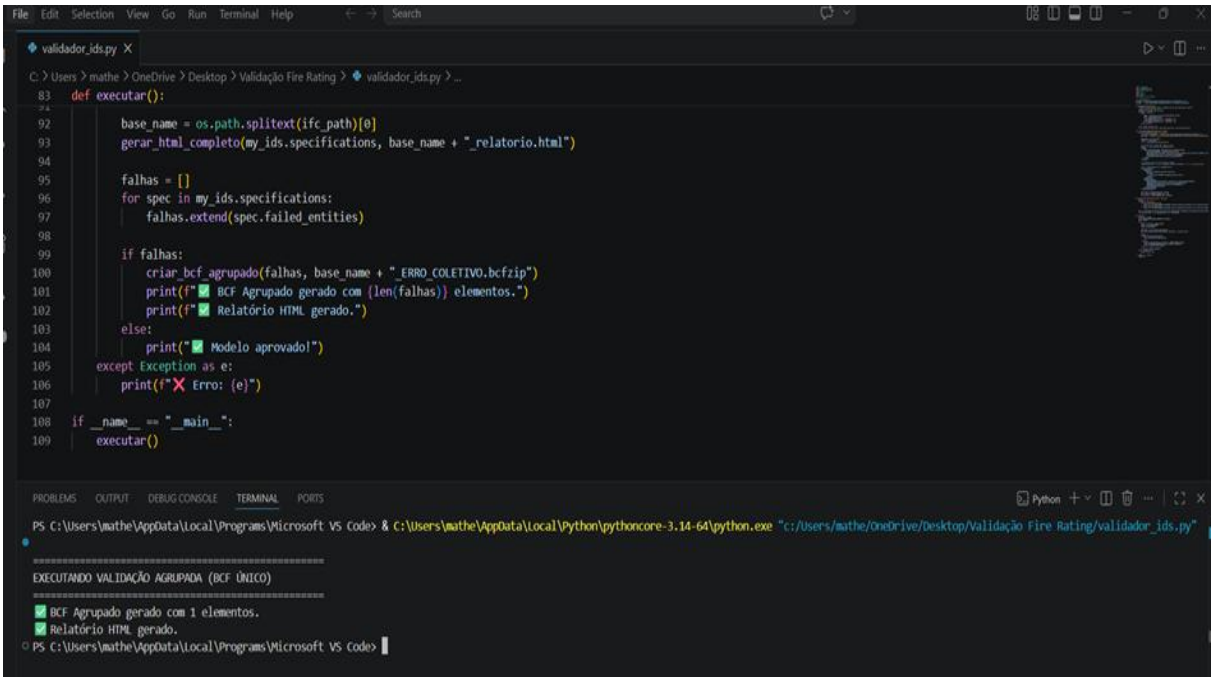
No experimento em questão, o algoritmo foi programado para auditar a conformidade da propriedade *Fire Rating* dentro do conjunto de propriedades padrão do IFC (*Pset_WallCommon*). Esta etapa é crucial para a garantia da qualidade do modelo, pois elimina a subjetividade e os erros inerentes aos processos de conferência visual ou manual.

Para viabilizar essa verificação, o script percorre a árvore de produtos do IFC e acessa a API de propriedades do *IfcOpenShell*, isolando o mapeamento de *Property Sets*. O algoritmo busca o atributo vinculado ao GUID de cada parede e verifica se o valor associado corresponde às definições estabelecidas no arquivo IDS, como o operador "*Is Defined*", garantindo que a auditoria ocorra no nível de código.

Após a identificação das inconsistências, o script inicia a estruturação das saídas. O primeiro produto é um relatório em formato HTML, gerado por uma rotina que converte os objetos de erro em uma tabela estilizada. Esse documento associa cada falha ao identificador único do elemento, facilitando a conferência gerencial fora do ambiente de modelagem.

Simultaneamente, o algoritmo processa a criação do arquivo BCF para coordenação técnica. Nesta etapa, o script utiliza o módulo geométrico para extrair as coordenadas globais das paredes reprovadas. Com esses dados, o código define um ponto de vista específico e injeta uma instrução de sobreposição de cor no XML do BCF, garantindo que o erro seja visualmente destacado (Figura 7).

Figura 7: VS Code mostrando a execução do script Python e a confirmação da validação dos elementos.



Fonte: autores.

Ao carregar o arquivo final no software de autoria, o projetista não precisa buscar manualmente as paredes sem informação. O plugin de coordenação lê as coordenadas

injetadas pelo script Python e conduz a visão do usuário diretamente ao objeto. Esse fluxo fecha o ciclo entre a detecção automatizada e a correção imediata da informação no ambiente de projeto.

RESULTADOS

Com base na metodologia adotada, apresentam-se os resultados da verificação automatizada dos modelos BIM por meio do relatório HTML (Figura 8) e do arquivo BCF (Figura 9), confirmando o êxito na geração dos produtos previstos pelo script desenvolvido.

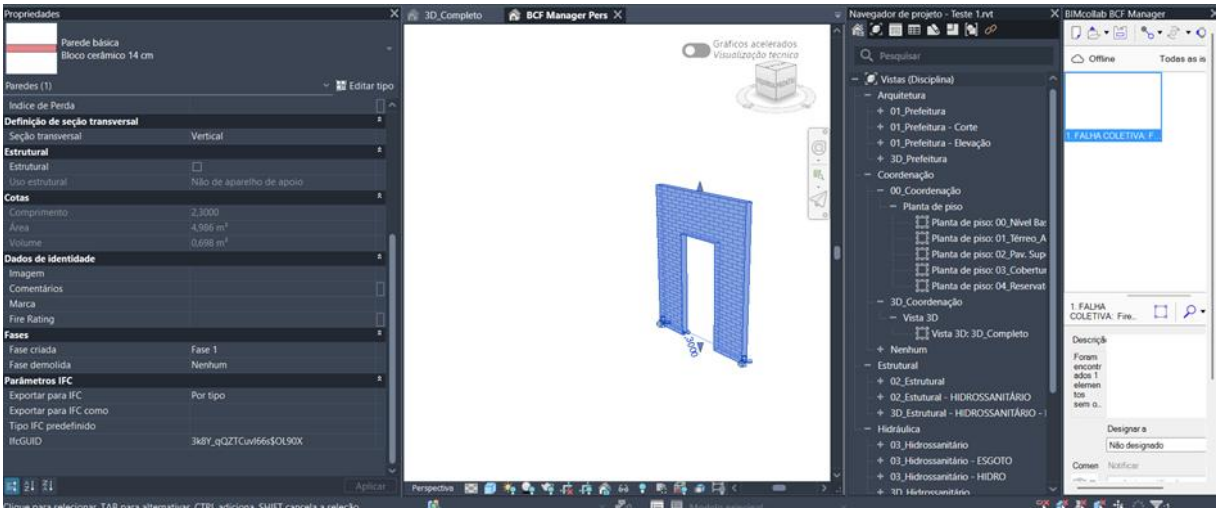
Figura 8: Relatório HTML.

Relatório de Auditoria

GUID	Classe	Status
3k8Y_qQZTCuvl66s$OL9HQ	IfcWall	OK
3k8Y_qQZTCuvl66s$OL90X	IfcWall	FALHA

Fonte: autores.

Figura 9: Identificação de não-conformidade informacional em elemento de parede, destacada visualmente via importação de arquivo BCF.



Fonte: autores.

A integração do padrão Information Delivery Specification (IDS) à biblioteca IfcOpenShell e à programação em Python viabiliza uma gestão da qualidade escalável e alinhada aos princípios do openBIM. Os resultados demonstraram a eficácia do fluxo de trabalho na detecção de omissões informacionais por meio da validação sintática, exemplificada pela propriedade *FireRating*, embora a verificação se limite à presença do dado, sem avaliar sua adequação normativa.

A utilização de scripts em Python permitiu realizar auditorias sem dependência de softwares proprietários, enquanto a IfcOpenShell possibilitou o acesso direto aos

dados do modelo IFC, reforçando o potencial do IDS como linguagem interoperável para verificação de requisitos informacionais.

Além disso, a geração automatizada de arquivos BCF contribuiu para a correção das inconsistências, permitindo isolar, destacar e geolocalizar elementos não conformes. A adoção de ferramentas de código aberto também evidenciou ganhos de autonomia, escalabilidade e redução de custos, sem comprometer a integridade dos dados.

DECLARAÇÃO SOBRE O USO DE INTELIGÊNCIA ARTIFICIAL

Durante a preparação deste artigo, todos os autores utilizaram a ferramenta Chat GPT Open AI, para a revisão e melhoria de coesão do texto. Após a utilização da ferramenta, os autores revisaram e editaram o conteúdo conforme necessário e assumem total responsabilidade pelo conteúdo do artigo publicado.

CONCLUSÃO

A crescente digitalização da indústria da construção tem ampliado a necessidade de mecanismos capazes de garantir a conformidade das informações presentes nos modelos BIM. Nesse contexto, a automação da verificação informacional surge como uma estratégia para aumentar a consistência e a confiabilidade dos modelos digitais.

Este estudo teve como objetivo investigar como requisitos de informação podem ser verificados automaticamente em modelos BIM por meio do padrão Information Delivery Specification (IDS). Para isso, foi desenvolvido um fluxo de trabalho baseado na integração entre IDS, IFC e ferramentas de código aberto, utilizando a biblioteca IfcOpenShell e rotinas em Python.

Os resultados demonstraram que o IDS permite estruturar requisitos de forma legível por máquina e verificar automaticamente a conformidade de dados em modelos IFC. A aplicação desenvolvida identificou a ausência da propriedade *FireRating* e gerou relatórios HTML e arquivos BCF para apoiar a correção das inconsistências encontradas, evidenciando a viabilidade do fluxo proposto.

Entretanto, a validação realizada neste trabalho foi conduzida a partir de um único estudo aplicado e de uma única regra de verificação. Assim, embora os resultados demonstrem a efetividade da abordagem, não permitem generalizar o IDS como base para a confiabilidade dos modelos BIM. Além disso, o experimento restringiu-se à validação sintática da informação.

Como trabalhos futuros, recomenda-se a aplicação do método em modelos e disciplinas distintas, contemplando múltiplos requisitos informacionais, bem como sua integração a verificações semânticas e normativas. Tais estudos poderão ampliar a

compreensão das potencialidades e limitações do IDS na gestão da informação em BIM.

REFERÊNCIAS

[1] CICCONE, A.; CIOTTA, V.; ASPRONE, D. **Integration Of Structural Information Within A Bim-Based Environment For Seismic Structural E-Permits.** Journal of Civil Engineering and Management, [s. l.], vol. 29, no 2, p. 171–193, 2023.

[2] BRASIL. **Decreto nº 10.306, de 2 de abril de 2020**. Estabelece a utilização do Building Information Modelling na execução direta ou indireta de obras e serviços de engenharia realizada pelos órgãos e pelas entidades da administração pública federal. Brasília, DF: Presidência da República, 2020.

[3] NAPPS, D.; AZIZ, A.; KÖNIG, M. **Digitalizing workplace safety regulations for rule-based validation of escape routes and movement areas in BIM**. Automation in Construction, vol. 186, p. 106883, 2026.

[4] EASTMAN, C.; TEICHOLZ, P.; SACKS, R.; LISTON, K. **BIM Handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors**. 2. ed. Nova Iorque: John Wiley & Sons, 2011.

[5] MA, Z. et al. **Automatic compliance checking of BIM models against quality standards based on ontology technology.** Automation in Construction, [s. l.], vol. 166, p. 105656, 2024.

[6] SUCCAR, B.; POIRIER, E. **Lifecycle information transformation and exchange for delivering and managing digital and physical assets.** Automation in Construction, [s. l.], vol. 112, p. 103090, 2020.

[7] GODAGER, B. et al. **Towards an improved framework for enterprise BIM: the role of ISO 19650.** Journal of Information Technology in Construction (ITcon), [s. l.], vol. 27, p. 1076, 2022.

[8] SALZANO, A. et al. **Existing assets maintenance management: Optimizing maintenance procedures and costs through BIM tools.** Automation in Construction, [s. l.], vol. 149, p. 104788, 2023.

[9] SUDAKOVA, K.; REMES, J.; TICHA, A. **Building Execution Plan as an effective document for Building Information Modelling.** Procedia Computer Science, [s. l.], vol. 239, p. 556–562, 2024.

[10] FISCHER, S. et al. **Extending Information Delivery Specifications for digital building permit requirements.** Developments in the Built Environment, [s. l.], vol. 20, p. 100560, 2024.

[11] TSAY, G. S. et al. **BIM for FM: understanding information quality issues in terms of compliance with owner's Building Information Modeling Requirements.** Frontiers in Built Environment, [s. l.], vol. 9, 2023.

[12] GRAGNANIELLO, C. et al. **BIM-based design and setup of structural health monitoring systems.** Automation in Construction, [s. l.], vol. 158, 2024.

[13] LIU, Y. et al. **Blockchain-enabled platform-as-a-service for production management in off-site construction design using openBIM standards.** Automation in Construction, [s. l.], vol. 164, p. 105447, 2024.

[14] LIU, H. et al. **Decoupling IFC models for reliable modification of shared references.** Automation in Construction, [s. l.], vol. 181, p. 106670, 2026.

[15] ZHANG, R.; EL-GOHARY, N. **Transformer-based approach for automated context-aware IFC-regulation semantic information alignment.** Automation in Construction, [s. l.], vol. 145, 2023.

[16] KREMER, N.; BEETZ, J. **Extending - information delivery specification - for linking distributed model checking services.** Proceedings of the European Conference on Computing in Construction, [s. l.], vol. 4, p. 0–0, 2023.