

AUTOMATIZAÇÃO EM PROJECTO DE SCIE: UM FLUXO DE TRABALHO BIM POTENCIADO POR PYTHON, DYNAMO E INTELIGÊNCIA ARTIFICIAL

Tiago Caferra ⁽¹⁾

(1) WALLNUT, Setúbal, 0009-0000-2428-6697

Resumo

A elaboração de projetos de Segurança Contra Incêndio em Edifícios (SCIE) em ambiente BIM enfrenta o desafio contínuo de conciliar a modelação tridimensional com uma vasta e rígida componente de cálculo regulamentar. Este artigo apresenta uma metodologia inovadora desenvolvida para Autodesk Revit que integra scripts Python (via extensão *pyRevit*), rotinas Dynamo e ferramentas nativas para automatizar processos prescritivos e gráficos. O fluxo de trabalho assenta num *template* estruturado que, através de algoritmos personalizados, classifica utilizações, calcula efetivos, determina cargas de incêndio e define Categorias de Risco automaticamente a partir de compartimentos virtuais (*Spaces*). Complementarmente, o sistema automatiza a representação simbólica 2D de equipamentos em conformidade normativa, contornando limitações gráficas nativas do software. Uma característica distintiva desta investigação é o recurso a Grandes Modelos de Linguagem (LLMs) como assistentes na escrita do código, democratizando o acesso à programação avançada. A aplicação prática demonstra que, ao transpor a regulamentação para algoritmos, o projetista garante a consistência bidirecional dos dados, assegurando resiliência a alterações arquitetónicas e elevando a coordenação interdisciplinar na fase de execução.

1. Introdução

A indústria da Arquitetura, Engenharia e Construção (AEC) atravessa uma profunda transformação digital, sendo o Building Information Modelling (BIM) o standard para a gestão integrada de informação [1]. Em Portugal, contudo, a adoção do BIM no projeto de Segurança Contra Incêndio em Edifícios (SCIE) colide com a complexidade do Regime Jurídico de SCIE (RJ-SCIE) [5] e do Regulamento Técnico (RT-SCIE) [6] e com a ausência de localização normativa nas plataformas de autor. Ao contrário de disciplinas regidas por normas internacionais, integradas nativamente no software, o projetista de SCIE depara-se com

ferramentas neutras face à legislação nacional, exigindo parametrizações manuais extensivas para evitar que o modelo se reduza a mera geometria tridimensional.

Consequentemente, o fluxo de trabalho mantém-se fragmentado: o cálculo prescritivo (frequentemente em Excel) dissocia-se da modelação geométrica (Revit), tornando as alterações arquitetónicas processos morosos e suscetíveis a erro. Acresce a dificuldade na produção de peças desenhadas simbólicas, dado que o Revit não foi concebido para gerir grafismos normativos bidimensionais complexos, perpetuando a dependência de soluções externas ou ajustes manuais.

Para internalizar o motor regulamentar no ambiente de modelação, a customização via API apresenta-se como solução lógica. Se historicamente a programação (Python, C#, Dynamo) constituiu uma barreira à adoção, a emergência da Inteligência Artificial Generativa (GenAI) alterou este paradigma. Atuando como “co-piloto”, a IA apoia a transposição da lógica regulamentar para algoritmos, democratizando o acesso à customização avançada.

O presente artigo propõe uma metodologia híbrida que integra a verificação regulamentar diretamente no modelo BIM. Recorrendo a scripts Python (via pyRevit) para processamento de dados e modelação de compartimentação, e a rotinas Dynamo para automatização gráfica, demonstra-se como a internalização algorítmica do RJ-SCIE promove consistência de dados, resiliência a alterações arquitetónicas e maior fiabilidade no desenvolvimento do projeto.

2. Enquadramento Teórico

A integração de metodologias BIM na verificação de conformidade regulamentar, enquadrada internacionalmente no conceito de Automated Code Checking (ACC), tem sido alvo de ampla investigação. Contudo, a sua aplicação prática em gabinetes de projeto permanece limitada. A verificação automática de regulamentos de segurança contra incêndio enfrenta desafios particulares devido à natureza híbrida das normas, que intercalam regras prescritivas quantitativas (ex.: larguras de vias de evacuação, distâncias a percorrer) com requisitos qualitativos sujeitos a interpretação subjetiva [2].

Embora existam soluções comerciais consolidadas para a verificação de normas (como o Solibri Model Checker), estas operam, por norma, sobre ficheiros exportados em formato Industry Foundation Classes (IFC). Esta abordagem post-modelling cria uma desconexão com o modelo de autor nativo. No contexto específico do regulamento português (RJ-SCIE), a inexistência de rulesets pré-definidos nestas plataformas obriga o projetista a uma parametrização manual morosa, diluindo os ganhos de produtividade esperados. Assim, a abordagem mais consequente passa por integrar a verificação diretamente no software de modelação (Autodesk Revit), garantindo um feedback contínuo durante a fase de conceção.

Para personalizar as plataformas BIM e automatizar tarefas, a linguagem de programação visual Dynamo consolidou-se como o padrão da indústria, beneficiando de uma interface gráfica intuitiva e de uma baixa barreira de entrada. Não obstante, à medida que os algoritmos de SCIE escalam em complexidade — exigindo iterações sobre centenas de compartimentos, verificações condicionais aninhadas e manipulação massiva de dados —, a programação visual revela-se ineficiente e de difícil manutenção, resultando no conhecido problema do “esparguete visual” [3].

Neste patamar de complexidade, a programação textual, nomeadamente através da linguagem Python, emerge como uma alternativa mais robusta. A extensão pyRevit permite a execução de

scripts Python diretamente na interface do Revit, conferindo um acesso profundo à sua Application Programming Interface (API). O Python assegura não só um processamento computacional mais célere e uma arquitetura lógica mais limpa, como possibilita a integração de bibliotecas externas de cálculo e análise de dados, cruciais para o dimensionamento em SCIE.

Historicamente, a transição do Dynamo para o Python representava um obstáculo severo para engenheiros e arquitetos sem formação em ciências da computação. Todavia, a democratização de Grandes Modelos de Linguagem (LLMs) e de assistentes de código baseados em Inteligência Artificial Generativa subverteu este paradigma. Estudos recentes [4] corroboram que a utilização de IA na programação reduz drasticamente o tempo de desenvolvimento. No contexto deste trabalho, a IA não atua como autora do projeto, mas sim como um "co-piloto" técnico: o engenheiro domina e define a lógica regulamentar (o "quê"), enquanto a IA agiliza a sintaxe do código Python (o "como"). Esta simbiose capacita os projetistas de SCIE a desenvolverem as suas próprias ferramentas de alta complexidade, competência outrora exclusiva de engenheiros de software.

3. Metodologia

A metodologia desenvolvida baseia-se na criação de um ecossistema digital no *software* Autodesk Revit, interligando a modelação geométrica com a lógica regulamentar do RJ-SCIE. O fluxo de trabalho foi estruturado em três eixos fundamentais: (1) Estruturação de Dados (*Template*), (2) Processamento Algorítmico (Python/Dynamo) e (3) Automatização Gráfica. O sistema opera integralmente dentro do ambiente nativo, garantindo a bidirecionalidade dos dados e evitando a exportação para plataformas externas. A arquitetura proposta recorre ao *addin pyRevit* para a execução de *scripts* Python através de uma interface personalizada (*Ribbon*), reservando o Dynamo para tarefas pontuais de manipulação geométrica visual.

3.1. Mapeamento Lógico e Estruturação de Dados

A transição de um processo tradicional (CAD/Excel) para um fluxo de trabalho BIM exigiu um mapeamento prévio exaustivo. Numa primeira fase, foram desenvolvidos fluxogramas de decisão condicional para a classificação dos Locais de Risco (Figura 1) e categorização de risco das Utilizações-Tipo (UT). Cada nó de decisão foi traduzido num parâmetro ou característica geométrica a ser incorporada nos espaços analisados, fornecendo a base lógica para a construção dos algoritmos.

A consolidação do processo exigiu a criação de um ficheiro de modelo (*Template*) normalizado. Este *template* integra um conjunto de Parâmetros Partilhados (*Shared Parameters*) associados, maioritariamente, à categoria *Spaces* (ex.: SCIE_UtilizacaoTipo, SCIE_DensidadeCarga, SCIE_EfetivoCalculado). Estes parâmetros atuam como recetores da informação processada, garantindo a persistência dos dados no modelo. Complementarmente, foram desenvolvidas famílias de *Generic Annotations* paramétricas. Esta categoria assegura total operabilidade com os algoritmos Python, recorrendo à gravação do ID do elemento modelado num parâmetro da própria anotação para garantir a sua rastreabilidade e evitar duplicações durante iterações de projeto.

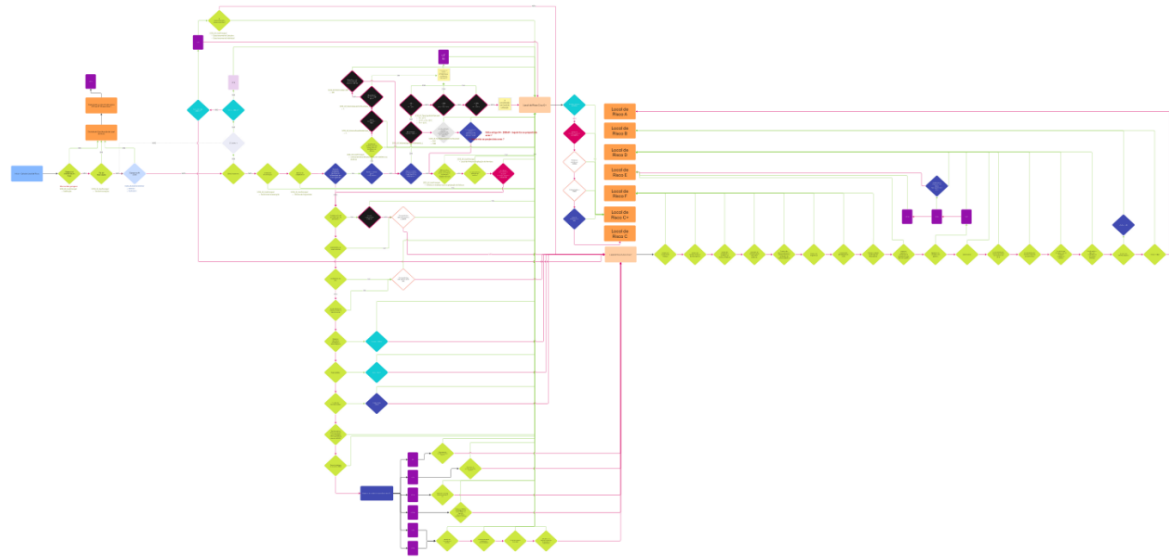


Figura 1: Tradução da lógica prescritiva do RJ-SCIE em fluxograma, sendo a base da engenharia de *prompts*.

3.2. Processamento Algorítmico e Automatização

As rotinas desenvolvidas podem ser agrupadas em três categorias funcionais: (i) gestão e processamento de dados regulamentares, (ii) geração automatizada de elementos construtivos e (iii) automatização gráfica bidimensional:

- **Classificação e Leitura de Espaços:** Leitura automática dos *Spaces*, interpretação de parâmetros e inferência de classificações funcionais através de lógica condicional. Integração com *Key Schedules* para associação de índices normativos.
- **Motor de Cálculo de Efetivos:** Processamento dos índices de ocupação regulamentares, efetivos pontuais e condições de risco agravado, devolvendo efetivo total, efetivo de evacuação e dimensionamento de saídas por espaço.
- **Agregação por Utilização-Tipo (UT):** Consolidação dos dados espaciais à escala da UT, incluindo determinação automática de altura útil e pisos abaixo do plano de referência.
- **Cálculo de Carga de Incêndio (Espaço e UT):** Automatização dos métodos determinísticos ou probabilísticos com base em parâmetros de atividade, área e fatores de combustibilidade, com posterior agregação à escala da UT.
- **Determinação de Categoria de Risco:** Avaliação automática das UTs face ao RJ-SCIE, atribuindo categoria individual e global do edifício.
- **Classificação de Locais de Risco:** Algoritmo condicional que cruza efetivos, cargas de incêndio e parâmetros técnicos para classificar cada compartimento segundo as tipologias regulamentares.
- **Modelação Automática de Compartimentação (REI):** Análise de adjacências tridimensionais entre *Spaces* e geração paramétrica de elementos construtivos (*Walls/Floors*) com resistência ao fogo adequada.
- **Geração de Space Separators:** Criação automática de limites espaciais independentes do modelo arquitetónico, assegurando controlo autónomo dos Locais de Risco.

- **Conversão de Path of Travel em Simbologia Normativa:** Transformação de percursos de evacuação em *Generic Annotations* conformes às normas gráficas.
- **Representação Bidimensional Automatizada (Dynamo):** Inserção automática de simbologia 2D associada a equipamentos 3D, com rastreabilidade bidirecional através de ID partilhado.

Tabela 1: Resumo funcional das rotinas desenvolvidas

	Gestão da Informação	Criação de Elementos Gráficos	Criação de Elementos (Modelo)
Tipo de Efetivo	X		
Cálculo Efetivo	X		
Dados UT	X		
Carga Incêndio Espaço	X		
Carga Incêndio UT	X		
Categoria Risco	X		
Local de Risco	X		
Modelação REI			X
<i>Space Separator</i>			X
Setas Evacuação		X	
Representação Equip.		X	
Represent. Sinalética		X	

```

elif ut_key == 11:
    if h <= 9 and pisos_abaixo == 0 and ef_total <= 100 and qs <= 5000: return "1ª"
    if h <= 28 and pisos_abaixo <= 1 and ef_total <= 500 and qs <= 50000: return "2ª"
    if h <= 28 and pisos_abaixo <= 2 and ef_total <= 1500 and qs <= 150000: return "3ª"
    return "4ª"
elif ut_key == 12:
    if "exterior" in local.lower():
        if qs <= 1000: return "1ª"
        if qs <= 10000: return "2ª"
        if qs <= 30000: return "3ª"
        return "4ª"
    else:
        if qs <= 500 and pisos_abaixo == 0: return "1ª"
        if qs <= 5000 and pisos_abaixo <= 1: return "2ª"
        if qs <= 15000 and pisos_abaixo <= 1: return "3ª"
        return "4ª"

```

Figura 2: *Excertos do código Python ilustrando a transposição sistemática da regulamentação para lógica computacional.*

3.3. O Papel da Inteligência Artificial e o Modelo de Maturidade

Dada a acentuada curva de aprendizagem associada à escrita de código para a API do Revit, o desenvolvimento destas rotinas incorporou o recurso a assistentes de IA (LLMs). A estratégia baseou-se na formulação de *prompts* iterativos para a geração da estrutura das funções, evidenciando o potencial da IA na democratização da programação para engenheiros.

O desenvolvimento destas ferramentas priorizou a automatização de tarefas de classificação repetitivas, cálculos aritméticos, verificações condicionais e representação gráfica. Este processo evolutivo de transição de um *workflow* CAD para BIM pode ser sistematizado num modelo de maturidade em quatro fases:

- **Fase 1 (CAD/Excel):** Fluxo totalmente segmentado sem automatização.
- **Fase 2 (BIM Incipiente):** Modelo 3D desconectado do cálculo; representação gráfica desvinculada dos elementos físicos.
- **Fase 3 (Atual):** Integração via Python e Dynamo. Os algoritmos alimentam parâmetros, e os parâmetros guiam a representação gráfica.
- **Fase 4 (Futura):** Expansão da lógica algorítmica para permitir o posicionamento espacial automático e otimizado de equipamentos de segurança e sinalética no modelo 3D, com base em regras de adjacência topológica, eliminando a intervenção manual remanescente.

4. Aplicação e Caso de Estudo

Para validar a exequibilidade e a resiliência da metodologia proposta, o fluxo de trabalho tem vindo a ser aplicado a múltiplos casos práticos em ambiente real de projeto. As iterações sucessivas permitiram a otimização contínua dos algoritmos. O processo-tipo de aplicação do sistema num projeto de SCIE descreve-se de seguida.

4.1. Aplicação do Fluxo de Trabalho

A implementação da metodologia obedece a uma sequência lógica rigorosa, demonstrando a interdependência entre a inserção de dados, o processamento analítico e a representação gráfica final.

4.1.1. Fase de Estudo Prévio ou Licenciamento

1. Definição Espacial e Inserção Preliminar de Dados: Numa primeira etapa, o projetista cria a categoria *Spaces* no Revit, gerando-os automaticamente a partir das *Rooms* previamente definidas no modelo de arquitetura linkado. De seguida, o utilizador introduz os dados preliminares necessários ao arranque do processo — fundamentalmente, a conceptualização do número de Utilizações-Tipo (UT) do edifício, a associação de cada espaço à respetiva UT e a sua classificação funcional primária. A maioria destas operações de inicialização é executada através de uma tabela de controlo (Schedule) pré-configurada no *template*.

2. Processamento Algorítmico (Python): Executam-se sequencialmente as rotinas desenvolvidas em *pyRevit*. Os algoritmos leem as áreas e usos preliminares, cruzam a informação com as lógicas do RJ-SCIE embutidas no código e preenchem automaticamente a totalidade dos parâmetros de *output* (ex.: SCIE_LocalRisco, SCIE_Efetivo, SCIE_DensidadeCarga).

3. Validação Visual Contínua (*Color Schemes*): Imediatamente após a execução dos *scripts*, as plantas de piso são atualizadas através de *Color Schemes* nativos do *template*. Esta codificação cromática automática permite ao projetista identificar instantaneamente potenciais erros de classificação ou topologia (ex.: um local de risco "E" ou "F" isolado incorretamente), servindo como uma barreira inicial e visual de controlo de qualidade.

4. Modelação Automática de Compartimentação: Executa-se a rotina de modelação de elementos corta-fogo (paredes e pavimentos). O *output* geométrico gerado pelo código é posteriormente sujeito a verificação e validação manual por parte do projetista, assegurando a compatibilidade arquitetónica.

5. Verificação de Peças Desenhadas e Escritas: Validação das vistas e tabelas (geradas automaticamente pelo *template*) referentes ao mapeamento de locais de risco, categorias de risco, compartimentação (REI) e reação ao fogo dos materiais.

6. Modelação de Evacuação: Modelação manual dos caminhos de fuga com recurso à ferramenta nativa *Path of Travel*. Subsequente execução do *script* de geração automática das setas de evacuação normalizadas (em anotação 2D) ao longo dos referidos percursos.

7. Modelação de Equipamentos e Sinalética: Modelação tridimensional manual das famílias de equipamentos de segurança e sinalética necessários à solução de SCIE (extintores, carretéis, detetores, etc.), posicionados criteriosamente pelo projetista.

8. Representação Simbólica e Rastreabilidade (*Dynamo*): Execução das rotinas visuais em *Dynamo*. O algoritmo lê os equipamentos e sinaléticas modelados (Passo 7) e insere as famílias de anotação 2D correspondentes na vista activa, garantindo o cumprimento estrito das normas gráficas e a legibilidade das peças desenhadas submetidas a licenciamento.

4.1.2. Fase de Projeto de Execução

Cumprido o licenciamento legal, o ecossistema BIM demonstra o seu valor acrescentado na transição fluida para a fase de execução, não exigindo a duplicação de modelos:

9. Matriz de Segurança: Desenvolvimento da matriz de segurança e a sua integração nas peças desenhadas.

10. Coordenação Interdisciplinar (*Clash Detection*): Aperfeiçoamento da coordenação tridimensional com as restantes especialidades (AVAC, Eletricidade, Estruturas). Os equipamentos de SCIE, por já estarem modelados com a sua geometria real (e não apenas como símbolos 2D), são alvo de *clash detection*, permitindo ajustes de localização e especificações técnicas precisas em fase de projeto.

11. Extração de Vistas para Obra: Geração de vistas tridimensionais (perspetivas e isometrias) e cortes detalhados focados nos equipamentos de SCIE coordenados, servindo como elementos de referência e clarificação técnica para as equipas de instalação em obra.

4.2. Evolução e Escalabilidade do Sistema

A aplicação prática do ecossistema a diversos cenários de projeto revelou a necessidade de ajustes iterativos, comprovando a resiliência e a escalabilidade da metodologia. Destacam-se dois exemplos paradigmáticos de otimização:

Refatorização da Lógica de Agrupamento de Utilizações-Tipo (UT): Nas versões inaugurais do algoritmo de classificação, a lógica de agrupamento assumia a existência de apenas uma UT de cada categoria (1 a 12) por edifício. Contudo, o contacto com casos de estudo mais complexos (ex.: edifícios multifuncionais com múltiplos blocos independentes) revelou a necessidade de coexistência de várias UTs da mesma tipologia num único projeto. Esta

limitação obrigou a uma refatorização do código Python. O algoritmo passou a não agrupar os espaços diretamente pela sua tipologia regulamentar, mas sim através de um identificador unívoco personalizado por projeto, garantindo o cálculo de cargas de incêndio e efetivos das UTs de forma estanque e correta, independentemente da complexidade arquitetónica.

Escalabilidade do Dicionário de Equipamentos (Dynamo): A arquitetura da solução foi pensada para garantir uma fácil expansão sem necessidade de reescrita de código. O *script* em Dynamo responsável pela inserção automática de simbologia 2D (*Generic Annotations*) assenta num sistema de "dicionário" partilhado. Quando surge a necessidade de incorporar um novo equipamento de SCIE não previsto inicialmente, o projetista necessita apenas de carregar a família 3D e a respetiva família de anotação 2D no modelo, atribuindo a ambas um código identificador idêntico num parâmetro de tipo específico. A rotina visual, ao ler este parâmetro partilhado, cruza automaticamente a geometria 3D com a simbologia 2D, assimilando o novo equipamento sem exigir qualquer alteração à lógica do algoritmo.

5. Resultados e Discussão

A análise comparativa entre o fluxo de trabalho tradicional (CAD/Excel) e a metodologia híbrida proposta revela ganhos substanciais, não apenas na mitigação do erro humano, mas sobretudo na consistência da informação ao longo do ciclo de vida do projeto.

5.1. Qualificação de Ganhos e Esforço Inicial

A aplicação prática do ecossistema a casos reais permitiu atestar de forma empírica ganhos processuais face aos métodos tradicionais. Num edifício de média complexidade, observou-se que a substituição do cálculo exaustivo manual por algoritmos altera fundamentalmente a natureza do trabalho. Mais do que uma simples redução cronometrável de tempo – difícil de isolar dado que o produto final BIM possui um nível de informação e coordenação interdisciplinar incomparavelmente superior ao CAD bidimensional –, o ganho traduz-se na eliminação de lapsos de transcrição de dados.

Não obstante estes ganhos na fase de cálculo, a adoção desta metodologia exige um esforço inicial de parametrização no software de autor consideravelmente superior ao método tradicional. A criação rigorosa de Spaces, a verificação de limites geométricos e a estruturação de dados de input obrigam a um investimento de tempo inicial (setup) que não existe no CAD bidimensional. Contudo, este custo temporal é largamente amortizado nas fases subsequentes: a automatização da compartimentação (REI), a inserção imediata de simbologia via Dynamo e, fundamentalmente, a capacidade de efetuar clash detection (compatibilização geométrica) nativa com as especialidades, conferem ao projeto de SCIE uma fiabilidade impossível de alcançar em ambientes segmentados.

5.2. Evolução da Maturidade BIM em SCIE

A investigação permitiu consolidar um modelo de evolução da maturidade BIM especificamente focado na disciplina de Segurança Contra Incêndio, mapeável em quatro estágios incrementais:

- Estágio 1 (Tradicional): Segmentação total entre a representação gráfica (CAD) e os modelos de cálculo (Excel). Inexistência de automatização geométrica ou validação cruzada de dados, dependendo exclusivamente da revisão humana.

- Estágio 2 (BIM Incipiente): Transição para a modelação 3D da arquitetura e equipamentos de SCIE, mantendo-se, porém, o motor de cálculo externo. A representação nativa de simbologia normativa no Revit exige um esforço manual desproporcional para gerar peças desenhadas legíveis.
- Estágio 3 (Metodologia Atual): Integração bidirecional de dados via Python (pyRevit) e validação visual instantânea (Color Schemes). A automatização foca-se na internalização do motor de cálculo e na resolução gráfica bidimensional via Dynamo, assegurando que as peças desenhadas refletem, sem margem para erro, os metadados do modelo.

5.3. Limitações e Trabalho Futuro (Estágio 4)

Apesar da robusta automatização alcançada nas componentes de cálculo, compartimentação e representação gráfica, o fluxo de trabalho atual apresenta uma limitação intrínseca: o posicionamento espacial dos equipamentos de segurança (extintores, botoneiras, carretéis) e da respetiva sinalética permanece um processo manual. O sistema atual atesta a presença do equipamento, cruza a informação técnica e atribui a simbologia 2D correta de forma automática; contudo, a decisão heurística do local exato de instalação continua a depender da expertise e intervenção direta do projetista.

O desenvolvimento futuro desta investigação (Estágio 4) incidirá sobre a exploração de algoritmos de Generative Design e otimização topológica. O objetivo é criar rotinas que, avaliando as distâncias de percurso reais (Path of Travel), raios de ação normativos e constrangimentos arquitetónicos, sugiram autonomamente a localização tridimensional ótima dos equipamentos, contribuindo para fechar o ciclo de automatização do projeto de SCIE.

5.4. Limitações Metodológicas, Manutenção e Interoperabilidade

A presente investigação encontra-se numa fase inicial de validação metodológica em contexto real de projeto. A quantificação objetiva de ganhos de produtividade face ao método tradicional não foi ainda formalmente aferida, dada a inexistência de práticas consolidadas de SCIE em BIM que permitam comparação direta e a variabilidade associada à maturidade dos modelos arquitetónicos. Acresce que a comparação linear de tempos de execução entre um projeto CAD 2D e um modelo BIM 3D coordenado não reflete o valor acrescentado da metodologia, tratando-se de produtos finais com níveis de fiabilidade distintos. O foco desta etapa incidiu, assim, na verificação da viabilidade técnica da internalização do motor regulamentar no Revit e na análise da consistência do sistema perante alterações arquitetónicas iterativas.

Quanto à manutenção normativa, a metodologia assenta numa arquitetura algorítmica modular em Python, permitindo que futuras revisões do RJ-SCIE sejam incorporadas através da atualização controlada das condicionantes lógicas, sem comprometer o template de dados nem o ecossistema BIM desenvolvido.

Relativamente à interoperabilidade openBIM, os parâmetros regulamentares encontram-se estruturados como Shared Parameters persistentes, constituindo base adequada para futuro mapeamento para entidades IFC (ex.: IfcSpace) e definição de Property Sets específicos de SCIE. A formalização desse mapeamento representa uma linha de desenvolvimento futura.

6. Conclusão

A fricção histórica entre a adoção BIM e o projeto de SCIE em Portugal — exacerbada pela ausência de localização de software e pelas exigências de representação bidimensional — pode ser superada através de customização algorítmica. A metodologia híbrida apresentada, ao internalizar o motor de cálculo do RJ-SCIE no Revit (via pyRevit) e ao resolver limitações gráficas com Dynamo, mitiga o risco de erro humano e elimina a dependência de folhas de cálculo externas, promovendo a consistência bidirecional dos dados ao longo do ciclo de projeto.

Um contributo central deste trabalho reside na demonstração da Inteligência Artificial Generativa como catalisador tecnológico na democratização da programação aplicada à AEC. Ao assumir o papel de “co-piloto” na escrita de código, a IA reduziu a barreira de entrada à customização avançada da API, capacitando projetistas a desenvolverem ferramentas algorítmicas próprias. Conclui-se que a automatização do SCIE não visa substituir a expertise do engenheiro, mas sim realocar o seu tempo — anteriormente dedicado à transcrição e verificação manual de dados — para a conceção e análise crítica de soluções de segurança.

O desenvolvimento futuro, orientado para modelos de otimização topológica e Generative Design, não se limitará ao posicionamento automatizado de equipamentos, mas procurará consolidar um quadro metodológico para a transposição sistemática de regulamentação prescritiva para sistemas algorítmicos em ambiente BIM, abrindo caminho à formalização de métricas de desempenho e à definição de modelos de maturidade específicos para a verificação regulamentar automatizada.

Referências

- [1] C. M. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*. Hoboken, NJ, USA: Wiley, 2011.
- [2] R. Amor and J. Dimyadi, “The promise of automated compliance checking,” *Developments in the Built Environment*, vol. 5, Art. no. 100039, 2021, doi:10.1016/j.dibe.2020.100039
- [3] C. Preidel and A. Borrmann, “Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling,” in *Proc. 32nd Int. Symp. Automation and Robotics in Construction (ISARC)*, Oulu, Finland, 2015. [Online]. Available: https://www.iaarc.org/publications/2015_proceedings_of_the_32st_isarc_oulu_finland/automated_code_compliance_checking_based_on_a_visual_language_and_building_information_modeling.html
- [4] C. Du, S. Esser, S. Nousias, and A. Borrmann, “Text2BIM: Generating Building Models Using a Large Language Model-Based Multiagent Framework” *Journal of Computing in Civil Engineering*, vol. 40, no. 2, 2026, doi:10.1061/JCCEE5.CPENG-6386
- [5] República Portuguesa, *Decreto-Lei n.º 220/2008, Regime Jurídico da Segurança Contra Incêndios em Edifícios*, Diário da República, 1.ª série, n.º 220, 12 Nov. 2008.
- [6] República Portuguesa, *Portaria n.º 1532/2008, Regulamento Técnico de Segurança contra Incêndio em Edifícios*, Diário da República, 1.ª série, n.º 250, 29 Dez. 2008.