INSTITUTO FEDERAL DE SANTA CATARINA

JEAN CARLOS TRICHES

FAMÍLIAS E *PLUG-INS* DE *LIGHT STEEL FRAME* PARA O *AUTODESK REVIT*

São Carlos-SC

2022

JEAN CARLOS TRICHES

FAMÍLIAS E *PLUG-INS* DE *LIGHT STEEL FRAME* PARA O *AUTODESK REVIT*

Monografia apresentada ao Curso de Engenharia Civil do Campus São Carlos do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenharia Civil

Orientadora: Ana Paula Antonello Sieg

São Carlos-SC

2022

Ficha de identificação da obra elaborada pelo autor.

Triches, Jean Carlos
FAMÍLIAS E PLUG-INS DE LIGHT STEEL FRAME PARA O AUTODESK **REVIT / Jean Carlos Triches; orientação de** Ana Paula Antonello Sieg. - São Carlos, SC, 2022.
77 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal de Santa Catarina, Câmpus São Carlos. Bacharelado **em Engenharia Civil. Departamento** Acadêmico **de Construção** Civil.
Inclui Referências.

1. Light Steel Frame. 2. BIM. 3. Famílias do Revit. **4. Plug-ins do Revit. I. Sieg, Ana Paula** Antonello. **II. Instituto Federal de Santa Catarina. III.** FAMÍLIAS **E PLUG-INS DE LIGHT STEEL FRAME PARA O AUTODESK** REVIT.

JEAN CARLOS TRICHES

FAMÍLIAS E *PLUG-INS* DE *LIGHT STEEL FRAME* PARA O *AUTODESK REVIT*

Este trabalho foi julgado adequado para obtenção do título de Engenheiro Civil, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São Carlos, 05 de Dezembro de 2022.

_____

Profa. Ana Paula Antonello Sieg, Me.
Orientadora
Instituto Federal de Santa Catarina

_____

Profa. Rafaella Aline Lopes da Silva Neitzel, Me.
Instituto Federal de Santa Catarina

_____

Prof. Allan Guimarães Borçato, Esp.
Instituto Federal de Santa Catarina

Dedico este trabalho a minha família,
professores e colegas por terem me
acompanhado nesta jornada e por
terem me auxiliado, tantas vezes,
quando eu mais precisava.

AGRADECIMENTOS

Agradeço em primeiro lugar à Deus por ser a base das minhas conquistas.

Aos meus pais, Anair e João, por acreditarem em minhas escolhas, apoiando-me e esforçando-se junto a mim, para que eu suprisse todas elas.

À minha irmã Ciza, que me acompanhou de perto em todo este trajeto, compartilhando das mesmas dificuldades, apreensões e vitórias.

À professora Ana, pela dedicação em suas orientações prestadas na elaboração deste trabalho, me incentivando e colaborando no desenvolvimento das ideias. E também por me fazer entender a disciplina de Análise Estrutural.

À todos os professores que, com paciência e dedicação, me guiaram até aqui.

Alguns homens vêem as coisas como são, e dizem “Por quê?”
Eu sonho com as coisas que nunca foram e digo “Por que não?”
(George Bernard Shaw, 1891)

RESUMO

O *Light Steel Frame* (LSF) é um sistema construtivo racional, estruturado em perfis metálicos de aço galvanizado, conformados a frio, com fechamento de painéis, que vêm sendo cada vez mais empregado no Brasil. Já as ferramentas BIM (*Building Information Modeling*) permitem uma modelagem mais eficiente das edificações, pois aumentam e tornam mais efetivos os fluxos de informações de projeto. Porém, percebe-se no mercado poucas ferramentas que auxiliem na modelagem de estruturas que utilizam este sistema através de *software*s BIM. E aquelas disponíveis, possuem um custo de licença muito elevado. Dentre as soluções BIM, o *REVIT* da *AUTODESK* é amplamente utilizado no mercado nacional. Desta forma, a fim de disponibilizar uma ferramenta que auxilie os profissionais e acadêmicos da Engenharia Civil a utilizarem este método construtivo através de ferramentas BIM, pretende-se com este trabalho de pesquisa, desenvolver e avaliar a utilidade de um conjunto de *plug-ins* e famílias de modelagem de estruturas em *Light Steel Frame* (LSF) para utilização no *AUTODESK REVIT*. Para isso foi realizado, inicialmente, uma pesquisa bibliográfica acerca do método construtivo *Light Steel Frame* e sobre o desenvolvimento de famílias e *plug-in*s para o *software AUTODESK REVIT*. Com base nesse conhecimento, foram criados elementos construtivos paramétricos vinculados a estruturas em LSF e ferramentas de apoio à modelagem. Para tanto, foram descritas as ações realizadas para modelagem, parametrização e representação gráfica dos elementos, aludindo acerca dos processos criativos e tomadas de decisão para isso. Por fim foram apresentados os resultados obtidos pela utilização das soluções para avaliar a usabilidade destas famílias ao modelar painéis estruturais para paredes arquitetônicas. Assim, pode-se verificar que os objetivos foram atendidos.

Palavras-Chave: *Light Steel Frame*. BIM. Famílias do *REVIT*. *Plug-ins* do REVIT

ABSTRACT

The Light Steel Frame (LSF) is a rational construction system, structured in cold-formed galvanized steel metal profiles, with panel closure, which have been increasingly used in Brazil. BIM (Building Information Modeling) tools allow a more rational modeling of buildings, as they increase and make project information flows more effective. However, there are few tools on the market that help in the modeling of structures that use this system through BIM software. And those available have a very high license cost. Among the BIM solutions, *AUTODESK*'s *REVIT* is widely used in the national market. In this way, in order to provide a tool that helps Civil Engineering professionals and academics to use this constructive method through BIM tools, it is intended with this research work, to develop and evaluate the usefulness of a set of modeling families and plug-ins of Light Steel Frame (LSF) structures for use in *AUTODESK REVIT*. For this, initially, a bibliographic research was carried out about the Light Steel Frame construction method and about the development of families and plug-ins for *AUTODESK REVIT* software. Based on this knowledge, parametric constructive elements and auxiliar tools linked to structures in LSF were created. For that, the actions taken for modeling, parameterization and graphic representation of the elements were described, alluding to the creative processes and decision-making for this. Finally, the results obtained by using the solutions to evaluate the usability of these families when modeling structural panels for architectural walls were presented. Thus, it can be verified that the objectives were met.

Keywords: *Light Steel Frame*. BIM. *REVIT* Families. REVIT *Plug-Ins*

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

IBGE – Instituto Brasileiro de Geografia e Estatística

IFSC – Instituto Federal de Santa Catarina

LSF – *Light Steel Frame*

BIM – *Building Information Modeling* - Modelagem da informação da Construção

PET – Poli Tereftalato de Etila

EPS – *Expanded Plystyrene*– Poliestireno Expandido

NBR – Norma Brasileira

OSB – Oriented Strand Board – Painéis de tiras de madeira orientada

PVC – Polyvinyl Chloride - Policloreto de vinila

XPS – Poliestireno Extrudado

VUP – Vida Útil de Projeto

API – *Application Programming Interface* - Interface de Programação de Aplicação

SUMÁRIO

1 INTRODUÇÃO

No Brasil predominam os métodos de construções tradicionais - onde existe um elevado consumo de matéria prima e perdas, e as empresas e profissionais da área buscam novas formas de construir evitando o desperdício e os impactos ambientais.

O *Light Steel Frame* (LSF) é um sistema construtivo racional, estruturado em perfis metálicos de aço galvanizado conformados a frio, com fechamento de painéis pré-fabricados com materiais cimentícios, fibras ou gesso, dispensando, assim, tijolos e armações convencionais - logo se propõe a evitar o desperdício e a geração de resíduos de obra (CASTRO, 2005).

Algumas das vantagens do uso dessa técnica, como alta resistência, baixo peso, grande precisão dimensional e uso de materiais reciclados e recicláveis, contribuem para uma construção mais sustentável (GORGOLEWSKI, 2006).

Porém percebe-se no mercado poucas ferramentas que auxiliem na modelagem de estruturas que utilizam este sistema através de *software*s BIM (*Building Information Modeling*). E aquelas disponíveis, possuem um custo de licença muito elevado como a STRUCSOFT MWF Pro Metal (STRUCSOFT, 2022) e AGACAD Metal Framing Wall (AGACAD, 2022), com preços na faixa de milhares de Dólares.

As ferramentas BIM permitem uma modelagem mais racional das edificações pois aumentam e tornam mais efetivos os fluxos de informações, a interação entre os projetistas e construtores, o melhor controle de planejamento de execução e do orçamento, além da prévia detecção e redução das incompatibilidades entre projetos (CAMPESTRINI *et al.,* 2015).

As soluções BIM da *AUTODESK* são amplamente utilizadas no mercado nacional. De fato, o *software* BIM de modelagem 3D *REVIT* é o *software* deste segmento mais utilizado no Brasil e possui muitos recursos, dentre eles é capaz de: modelar componentes de construção, analisar e simular sistemas e estruturas; gerar documentação por meio de modelos; importar, exportar e vincular dados com os principais formatos da indústria; possibilitar o trabalho de modo colaborativo; comunicar a intenção do projeto de maneira mais eficaz para os seus proprietários e demais membros da equipe. Sendo que para isso, os usuários se utilizam de modelos - as chamadas "famílias", e *plug-ins* – que são módulos e extensões, para criar as plantas e visualizações 3D (GUALBERTO, 2021).

Logo, desenvolver ferramentas que auxiliem os profissionais da Engenharia Civil a utilizarem o método construtivo *Light Steel Frame* através de ferramentas BIM, torna-se relevante e justifica os esforços despendidos neste sentido.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver e avaliar a utilidade de um conjunto de famílias e *plug-ins* de modelagem de estruturas em *Light Steel Frame* (LSF) para utilização no *AUTODESK REVIT.*

1.1.2 Objetivos específicos

- Conhecer e apresentar o método construtivo *Light Steel Frame*;
- Compreender o desenvolvimento de famílias e de *plug-ins* no *AUTODESK REVIT*;
- Modelar, desenvolver e testar o conjunto de famílias e *plug-ins* para utilização de LSF no *AUTODESK REVIT*;
- Avaliar a utilidade e funcionalidade das famílias e *plug-ins* desenvolvidos;

2 DESENVOLVIMENTO

2.1 O Sistema Construtivo *Light Steel Frame*

No Brasil, como uma das alternativas à construção de alvenaria convencional tem-se o *Light Steel Framing* (LSF). É um sistema construtivo de concepção racional e estruturado em perfis metálicos de aço galvanizado conformados a frio e de painéis estruturais e não estruturais, dispensando, assim, tijolos e armações convencionais (SANTIAGO, 2008).

Como componentes de fechamento podem ser usadas as chapas de gesso acartonado, de PVC (Policloreto de vinila) rígido ou as cimentícias, conforme a diretriz técnica 3 do Sistema Nacional de Avaliações Técnicas (MINISTÉRIO DAS CIDADES, 2012).

Batista (2011) aponta que por ser um sistema construtivo altamente industrializado, o LSF proporciona uma maior eficácia e brevidade na execução das construções. Nas próximas seções busca-se compreender e explicar este processo construtivo.

2.1.1 Projeto e Processo Construtivo

O principal conceito estrutural que define o sistema LSF é dividir a estrutura em um grande número de elementos estruturais de modo que cada elemento resista a uma pequena fração da carga. Dessa forma, podem ser utilizados perfis mais finos e painéis mais leves e fáceis de manusear (ROVARIS, 2016).

Santiago (2012) enumera uma série de características do LSF que o distingue de outros sistemas construtivos tradicionais, dentre elas sua combinação de elementos estruturais, de isolamento, de acabamentos, de instalações, dentre outros com as mais diversas aplicações e funções. O autor chama a atenção para a série de vantagens deste sistema, como o aço ser um material de resistência comprovada e ter um ótimo controle de qualidade e conformidade na produção, permitindo assim uma maior precisão construtiva.

Além do aço ser um material acausto, ou seja, que não serve como combustível ao fogo, ele pode ser reciclado diversas vezes sem perder qualidade, é de fácil obtenção nos mais diversos tipos de perfis e fabricados a frio, tem uma excelente

durabilidade proporcionada pelo processo de galvanização e é um material de fácil transporte e manuseio. Ele permite a chamada construção a seco, o que diminui muito o desperdício e o uso de recursos naturais como a água potável que, no processo, só é utilizada para a fundação e no assentamento de revestimentos cerâmicos (SANTIAGO 2012).

Santiago (2008), destaca que a utilização de aço nos processos construtivos civis, devem preencher vários aspectos da construção industrializada, sendo que o processo de produção se baseia em elementos padronizados, que estão dispostos por uma lógica modular.

Logo o projeto deve tirar o máximo proveito destas características, orientando a equipe na gestão da obra, definindo precisamente o número de pilares, trilhos, parafusos e chapas, sempre considerando e buscando o menor peso estrutural, uma eficiente distribuição dos esforços, isolamento térmico e acústico, as particularidades técnicas e o tempo de execução e principalmente a redução do desperdício e do volume de resíduos (CASTRO, 2005).

2.1.2 Metodologias Construtivas

O *Light Steel Frame* possui três metodologias de construção: o método Stick, o método de painéis e o método modular. O método Stick é caracterizado pelo corte e montagem de todos os objetos no local da obra. Suas estruturas são formadas a partir de lajes, telhados e colunas; e os frames podem vir preparados para receber as instalações elétricas e hidráulicas. São vantagens desse método: a não necessidade de um local para a pré-fabricação dos perfis; as ligações dos elementos são de fácil execução; e a facilidade de transportes dos elementos (BEN, 2016).

O método de painéis, por outro lado, caracteriza-se pela pré-fabricação fora do canteiro de obras, porém a montagem continua sendo executada no local da obra. Dessa maneira, esta metodologia proporciona precisão, rapidez, qualidade, e redução do trabalho no canteiro uma vez que são pré-fabricados. Com relação aos demais processos e elementos da construção (subsistemas e demais elementos estruturais) são realizados por meio de parafusos, tendo um ganho considerável com a velocidade de montagem, controle de qualidade e precisão das medidas (SANTIADO, 2012).

Já o método modular é caracterizado por não haver nenhum trabalho no canteiro de obras, sendo o empreendimento entregue com todas as estruturas

montadas e os subsistemas já instalados (BEN, 2016).

2.1.3 Estrutura e fundação

LSF é o processo de armação de aço estrutural, formado por vários elementos individuais unidos entre si, que juntos passam a resistir e moldar as cargas exigidas da edificação (SANTIAGO, 2012). Segundo Gropius (2004), a estrutura é executada em perfis de aço galvanizado com espaçamento definido com base em cálculos estruturais e normas específicas quanto ao dimensionamento e forma da construção.

Como a concepção do LSF é voltado para fabricação e montagem industrializada, por padrão são utilizados, como montantes, perfis do tipo “U” padronizados para a composição dos painéis destinados à execução de paredes, pisos, telhados e fachadas com função estrutural (ROVARIS, 2016).

Para Castro (2005), devem-se utilizar contraventamentos mantendo-se o ângulo de inclinação de 30º a 60º entre as fitas metálicas e as bases dos painéis pois, conforme explica, deve-se evitar ângulos menores que 30 graus, pois apresentam baixo desempenho e a ocorrência de deformações.

O autor assinala que as placas de cobertura também são capazes de absorver parte das solicitações, como as cargas laterais. Jardim e Campos (2013), apresentam resultados experimentais neste sentido o que proporciona uma maior simplicidade no projeto e execução de painéis de parede e de pisos, pois é possível a eliminação de diagonais. Nesses casos é permitido a instalação de bloqueadores horizontais, ou seja, peças com perfis “U” que deverão ser instaladas perpendicularmente aos montantes.

Devido às baixas solicitações de carga de uma estrutura em *Light Steel Framing*, normalmente utiliza-se a fundação laje radier (SANTIAGO *et al*., 2012). Castro (2005), atenta para a necessidade da utilização de *parabolts* expansivos para a ancoragem da estrutura LSF à fundação de concreto. Essas ancoragens devem ser previstas em projeto e calculadas para resistir aos carregamentos verticais, em especial da ação dos ventos, garantindo também a estabilidade estrutural geral.

Quanto ao revestimento da estrutura de aço, a camada mínima definida na norma NBR 15253 (ABNT, 2014) para proteção do aço varia de 150 a 180 g/m² para perfis estruturais e 100 g/m² para não estruturais.

2.1.4 Painéis, aberturas e Revestimento

Os materiais mais empregados para o revestimento vertical das edificações construídas com o *Light Steel Frame* no Brasil, ou seja, as placas de cobertura estruturadas, são chamadas de OSB (*Oriented Strand Board*) e podem ser basicamente de dois tipos: placas cimentícias - normalmente utilizadas para fechamentos externos, e de gesso acartonado - para fechamentos internos (SANTIAGO, 2008). Ao contrário das chapas de compensado e MDF, as placas de OSB são construídas de forma a proporcionar resistência mecânica com fins estruturais (MASSISA, 2007).

As placas cimentícias são compostas de uma mistura homogênea de cimento Portland com polpa ou fibras minerais, celulose ou até mesmo de fios sintéticos como fibra de sílica. Estas placas podem ser utilizadas tanto para fechamento interno como externo dos painéis, horizontais ou verticais, tanto em áreas secas como úmidas ou expostas à chuva (VIVAN, 2011).

Já as placas de gesso acartonado são formadas basicamente pelo componente interno homogêneo de gesso hidratado e de revestimentos por lâminas de papel cartão, o que proporciona uma maior resistência aos esforços de tração, compressão e flexão, bem como à perfuração e rachamento. Esse tipo de placa é muito leve, deve ser utilizada apenas no revestimento interno dos painéis e não possui função estrutural. (VIVAN, 2011).

Estas placas são comercializadas em medidas padrões: sendo 1,20m de largura nominal e comprimento variando entre 1,80 e 3,60m. A espessura também é variável entre 9.5 e 15.5mm.

Para Santiago *et al.* (2012), o sistema de aberturas é executado de forma similar aos métodos ditos tradicionais de construção. Oliveira (2012), atenta para para o fato de que os painéis em que serão inseridas as esquadrias deverão possuir reforços que atuarão para transmitir os esforços até os elementos de fundação. Essas vergas e contra-vergas são ligadas a peças verticais chamadas ombreiras.

Por fim, Campos (2013), explica que o revestimento final sobre as placas é executado de forma similar ao das paredes em alvenaria. Já a utilização de placas rígidas de OBS para o fechamento de lajes e o seu revestimento deve considerar o tipo de aplicação.

2.1.5 Lajes

A estrutura de piso em LSF é composta por perfis de seção “U” dispostos na horizontal e com as almas alinhadas. A essas estruturas denominamos vigas de piso. No geral, as mesas dos perfis das vigas de piso têm a mesma dimensão das mesas dos montantes, porém a altura da alma dessas vigas é determinada em função do vão entre os apoios (SANTIAGO *et al.*, 2012).

Quanto aos painéis de fechamento, há também que se considerar se a laje é do tipo úmida ou seca. Para as lajes tipo seca utilizam-se placas OBS para a cobertura. Já para as lajes do tipo úmida utilizam-se chapas metálicas onduladas que são preenchidas com concreto que serve de base para o contrapiso. (SANTIAGO *et al.*, 2012).

2.1.6 Instalações Hidrossanitárias

Domarascki e Fagiani (2009) esclarecem que no sistema LSF, quanto às instalações hidrossanitárias, utiliza-se as mesmas técnicas e materiais dos sistemas construtivos tradicionais. Portanto as tubulações e equipamentos devem ser previstos em projeto. Quanto à execução, ela se torna mais fácil em virtude do vão interno das paredes e pelos furos dos montantes que podem ser pré-executados ou facilmente executados em obra. Também, nas áreas molhadas, pode-se utilizar placas comercialmente disponíveis, desenvolvidas especialmente para este fim.

2.1.7 Componentes de Vedação Termoacústica

Para a garantia do conforto termoacústico, deve-se utilizar um recheio de lã mineral ou pet entre as placas de revestimento. Este sistema tem um melhor desempenho se comparado ao da alvenaria tradicional (CARREGARI, 2019). Os materiais mais comuns para esta aplicação são as lãs, o EPS e os painéis de poliuretano.

As lãs podem ser de vidro, rocha ou PET. Nas paredes exteriores, a aplicação da lã é sempre feita pelo lado interno da construção. Já para a execução do EPS (Poliestireno Expandido), este é feito após a colocação do painel, pois o fechamento é utilizado como apoio para sua aplicação (CASTRO, 2005).

2.1.8 Cobertura

Para a estrutura da cobertura são utilizados os mesmos perfis do tipo “U” de aço galvanizados conformados a frio utilizados nas demais estruturas do LSF. É possível a execução das mais variadas formas de cobertura, desde planos horizontais até extremamente inclinados. E os tipos de revestimentos aplicáveis são os mesmos utilizados nas coberturas convencionais (CASTRO, 2005). O mesmo autor atenta para o fato de que a estrutura da cobertura deve suportar, além do seu próprio peso, o das telhas, dos forros, dos isolantes e das calhas, além de suportar os esforços gerados pela ação do vento e/ou do acúmulo de neve.

A forma mais comum são as coberturas estruturadas por treliças e tesouras já muito utilizadas com o emprego de madeira. No Brasil, as estruturas de cobertura em aço já vêm substituindo as treliças de madeira em função de sua grande resistência e leveza. Santiago *et al.* (2012), lembram que essas treliças podem vir pré-fabricadas ou serem montadas no canteiro de obras.

2.1.9 Instalações Elétricas

Assim como as instalações hidro sanitárias, as instalações elétricas devem ser previstas em projeto e executadas logo após a montagem das estruturas das paredes, lajes, coberturas e revestimentos externos, e antes da aplicação do isolamento termoacústico e do revestimento interno (GRUBLER, 2021).

Portanto, os materiais empregados e os princípios e considerações de projeto, são os mesmos das edificações executadas nas estruturas convencionais, seguindo-se sempre os requisitos das normas pertinentes.

2.2 Normatização sobre *Light Steel Framing* no Brasil

Em 23 de Maio de 2022, foi aprovada a NBR 16970 (ABNT, 2022), denominada como *Light Steel Framing*: Sistemas construtivos estruturados em perfis leves de aço formados a frio, com fechamentos em chapas delgadas - que regulamenta este tipo de sistema construtivo, sendo composta por três partes: 1 - Desempenho; 2 - Projeto estrutural; e 3 - Interface entre sistemas.

A NBR 16970 (ABNT, 2022) destina-se a utilização do sistema para edificações

residenciais e não residenciais de até dois pavimentos, especificando a qualidade mínima dos materiais e o desempenho esperado de cada parte do sistema, facilitando, também, o processo de financiamento junto a órgãos e instituições financeiras.

2.2.1 Desempenho

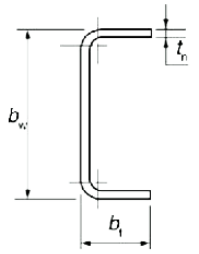
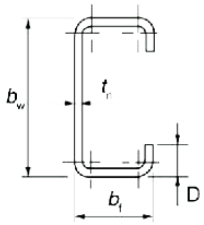
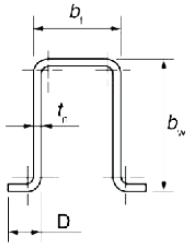
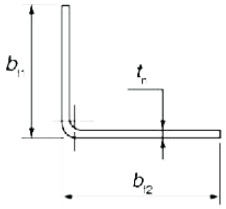
Sobre o desempenho, a NBR 16970 (ABNT, 2022) traz os requisitos de cada componente do sistema. Para isso, primeiramente são definidos os termos e materiais relacionados para então definir os requisitos dos materiais a serem utilizados, os requisitos de desempenho de cada sistema e referenciando outras normativas específicas.

O item 3.19.1, por exemplo, define guias e montantes como:

> perfis obtidos por dobramento em prensa dobradeira de tiras cortadas de chapas ou bobinas, ou por conformação contínua em conjunto de matrizes rotativas a partir de bobinas laminadas a frio ou a quente, sendo ambas as operações realizadas com aço em temperatura ambiente, utilizados na composição de elementos estruturais LSF (ABNT, 2022, p.21)

Já estes perfis com suas seções transversais, suas designações e formas de utilização estão definidos como apresentados na Figura 1. A norma designa os tipos: U simples para utilização como guia, ripa, bloqueador, sanefa ou terça; U enrijecido - ou Ue - para utilização como bloqueador, enrijecedor de alma, montante, verga, viga, terça e guia enrijecida; cartola para utilização como viga, ripa e terça; e a cantoneira de abas desiguais L.

Figura 1 - Tabela de perfis de aço da NBR 16970 (ABNT, 2022)

Seção transversal	Designação ABNT NBR 6355	Utilização
	U simples U $b_w \times b_f \times t_n$	Guia Ripa Bloqueador Sanefa Terça
	U enrijecido Ue $b_w \times b_f \times D \times t_n$	Bloqueador Enrijecedor de alma Montante Verga Viga Terça Guia enrijecida (sistema com encaixes estampados)
	Cartola Cr $b_w \times b_f \times D \times t_n$	Viga Ripa Terça
	Cantoneira de abas desiguais L $b_{f1} \times b_{f2} \times t_n$	Cantoneira

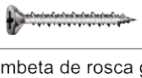
Fonte: ABNT (2022)

Quanto ao revestimento da estrutura de aço, ao contrário do definido anteriormente na norma NBR 15253 (ABNT, 2014), a camada mínima definida para proteção do aço agora deve variar de 275 a 350 g/m². Já a espessura nominal dos perfis (tn) para perfis U simples ou enrijecidos, cartola ou cantoneiras deve variar entre 0,8 e 3,0 mm e ser maior que 0,65 mm para perfil cartola usado como ripa.

A norma define também os elementos de fixação, ancoragens e seus suportes. Assim, os fixadores são utilizados para fazer a união entre os elementos construtivos, assegurando que estas não sofram deformações e rupturas. Devem ser de aço

carbono, cementado e temperado conforme exemplificados na Figura 2.

Figura 2 - Tabela de tipos de fixadores da NBR 16970 (ABNT, 2022)

Tipo de parafuso	Aplicação
Cabeça flangeada com ponta de broca  Cabeça sextavada com ponta de broca 	Parafusos aplicados entre perfis metálicos LSF de espessura igual ou superior a 0,80 mm
Cabeça flangeada com ponta de agulha 	Parafusos aplicados entre perfis metálicos LSF pré-furados de espessura igual ou superior a 0 80 mm
Cabeça chata dentada com ponta de broca e asas ou aletas 	Parafusos para fixação das vedações externas ou internas de alguns tipos de chapas de fibrocimento [a] em perfil de aço
Cabeça chata escariante com ponta de broca sem asas ou aletas 	Parafusos para fixação das vedações externas ou internas (OSB, cimentícia [a]) em perfil de aço
Cabeça trombeta com ponta de broca 	Parafusos para fixação das chapas *drywall* em perfil de aço
Cabeça chata dentada de rosca dupla (HI-LO) com ponta agulha  	Parafusos para fixação de chapa cimentícia [a] somente sobre a chapa OSB
Cabeça trombeta de rosca grossa com ponta agulha 	Parafusos para fixação de chapa *Drywall* foram desenvolvido para a aplicação em perfis de madeira
Cabeça sextavada flangeada com arruela de vedação fixa ou móvel, e ponta de broca 	Parafusos aplicados para fixação de telhas metálicas à estrutura da subcobertura.

Fonte: ABNT (2022)

Os requisitos para estes elementos de fixação são os de: Tempo mínimo de corrosão através de ensaio de névoa salina, segundo a NBR 8094 (ABNT, 1983), sendo de no mínimo 96 horas para fixação de chapas internas de fechamento aos quadros reticulados, 240 horas para ligações entre perfis metálicos para a fixação de quadros estruturais, 480 horas para fixação de chapas externas aos quadros estruturais e de 240 horas para chumbadores.

O poder de perfuração dos elementos de fixação deve ser maior que 1 s para ponta de agulha e de 4 s para ponta de broca. E a resistência à torção deve ser maior

que 4,7 N.m (ABNT, 2022).

Quanto aos componentes de fechamento e revestimento vertical por chapas de fibrocimento, estes devem atender ao estabelecido na NBR 15498 (ABNT, 2021a) e também ao definido nesta norma NBR 16970 (ABNT, 2022) conforme a Tabela 5, apresentada na Figura 3.

Figura 3 - Tabela de Requisitos para Chapas de Fibrocimento da NBR 16970 (ABNT, 2022)

Especificação	Requisito			Normas
Resistência mecânica mínima (Resistência à tração na flexão em MPa) (1)	Categoria	CLASSE A e B (condição saturada)	CLASSE C (condição ambiente)	ABNT NBR 15498
	1	-----	4	
	2	4	7	
	3	7	10	
	4	13	18	
	5	18	24	
Permeabilidade à água	No ensaio, podem aparecer traços de umidade na face inferior das placas, mas em nenhum caso deve haver formação de gotas de água nessa face. Isto não se aplica às placas com acabamento ou revestidas.			ABNT NBR 15498
Ensaios de resistência mecânica após envelhecimento acelerado por imersão e secagem	Nas placas ensaiadas conforme a norma de referência, o limite Li do resultado médio indicado deve ser superior a 0,70			ABNT NBR 15498
Ensaio de resistência mecânica após envelhecimento acelerado por água quente	Nas placas ensaiadas conforme a norma de referência, o limite Li do resultado médio indicado deve ser superior a 0,70			ABNT NBR 15498
Variação dimensional por imersão e secagem	Para juntas invisíveis em revestimentos aderidos, os valores de variação dimensional devem ser no máximo 2,5 mm/m. Para junta visível e revestimentos não aderidos, os valores de variação dimensional devem ser informados pelo fabricante nas especificações do produto, bem como a forma de instalação.			-----
NOTA As chapas categoria A são indicadas para aplicações externas, sujeitas à ação direta das intempéries, como sol, chuva, congelamento ou neve. As chapas categoria B são indicadas para aplicações externas, não expostas à ação direta de intempéries, podendo ficar expostas à umidade, calor e eventual congelamento. As chapas categoria C são indicadas para aplicações internas, como paredes internas, pisos, forros e substratos, podendo ficar exposta ao calor e à umidade, mas não a congelamento.				

Fonte: ABNT (2022)

As chapas de gesso devem atender às normas NBR 14715-1 (ABNT, 2021c) e NBR 15758-1 (ABNT, 2009), bem como o apresentado na tabela 6 da norma, apresentada, como exemplo, na Figura 4.

Figura 4 - Tabela parcial de Requisitos para Chapas de Gesso da NBR 16970 (ABNT, 2022)

<table>
<tr><th>Especificação</th><th colspan="3">Requisito</th><th>Normas com métodos de ensaio</th></tr>
<tr><td rowspan="5">Caracterização dimensional</td><td>Característica geométrica</td><td>Tolerâncias</td><td>Limite</td><td rowspan="5">EN 15283</td></tr>
<tr><td>Espessura</td><td>± 0,5 mm</td><td>-</td></tr>
<tr><td>Largura</td><td>+ 0 mm
– 4 mm</td><td>Máximo de 1 200 mm</td></tr>
<tr><td>Comprimento</td><td>+ 0 mm
– 5 mm</td><td>Máximo de 3 600 mm</td></tr>
<tr><td>Esquadro</td><td>≤ 2,5 mm</td><td>-</td></tr>
<tr><td rowspan="2">Resistência mecânica mínima (Resistência à tração na flexão em MPa) – Estado de equilíbrio</td><td>Espessura (mm)</td><td>Transversal</td><td>Longitudinal</td><td rowspan="2">EN 12467</td></tr>
<tr><td>t</td><td>$\frac{29,4}{t}$</td><td>$\frac{75,3}{t}$</td></tr>
<tr><td rowspan="2">Resistência mecânica mínima (Resistência à tração na flexão em MPa) – Estado saturado</td><td>Espessura (mm)</td><td>Transversal</td><td>Longitudinal</td><td rowspan="2">EN 12467</td></tr>
<tr><td>t</td><td>$\frac{20,6}{t}$</td><td>$\frac{52,7}{t}$</td></tr>
<tr><td rowspan="2">Ensaios de resistência mecânica após envelhecimento acelerado por imersão e secagem – 50 ciclos</td><td colspan="3">Nas chapas ensaiadas conforme a norma de referência, o limite Li do resultado médio indicado deve ser superior a 0.7 (comparação feita entre placas envelhecidas e placas saturadas antes do envelhecimento).</td><td rowspan="2">EN 12467</td></tr>
<tr><td colspan="3">A resistência mecânica das amostras envelhecidas (após 50 ciclos) é obtida após condicionamento em laboratório por sete dias (estado de equilíbrio após o envelhecimento).</td></tr>
<tr><td>Dureza</td><td colspan="3">O diâmetro máximo da depressão deve ser inferior a 15 mm.</td><td>EN 15283</td></tr>
</table>

Fonte: ABNT (2022)

Assim, também são apresentados requisitos para as placas de fechamento do tipo Chapa OSB (*oriented strand board*), painéis de compensado estrutural fenólico, *sidings* cimentícios, de PVC, *Basecoat* e EIFS - um sistema de revestimento e isolamento térmico exterior.

Os painéis de fechamento horizontais, ou seja, aqueles designados para laje seca ou foros, quer sejam de chapas de fibrocimento, chapas de OSB ou gesso para *drywall*, devem assegurar o atendimento aos requisitos de desempenho definidos na norma.

Já os elementos isolantes, distinguidos entre elementos isolantes de banda acústica, térmicos, termoacústicos, e de barreira de vapor e umidade, devem atender a diversos requisitos para cada tipo de material empregado. A banda acústica, que tem função de vedar juntas de interface entre diferentes subsistemas, deve atender o disposto na tabela 12 da norma, apresentada na Figura 5.

Figura 5 - Tabela de Requisitos para Banda Acústica da NBR 16970 (ABNT, 2022)

Especificação		Requisito	Métodos de ensaio
Tensão de ruptura (kN/m²)	Direção longitudinal	Mín. 200 kN/m²	ASTM D 412-16
	Direção transversal	Mín. 150 kN/m²	ASTM D 412-16
Alongamento (%)	Direção longitudinal	Mín. 70 %	ASTM D 412-16
	Direção transversal	Mín. 90 %	ASTM D 412-16
Resistencia à compressão	Compressão máx. 50 %	Mín. 80 kN/m²	ASTM D 375/D 375M-95
Absorção de água (7 dias)	< 10 %		ASTM D 375/D 375M-95
Densidade média	(30 a 46) kg/m³		ASTM D 375/D 375M-95

Fonte: ABNT (2022)

Para os elementos EPS e XPS, que são isolantes térmicos rígidos de poliestireno, estes devem atender a requisitos de absorção de água, variação de volume, estabilidade térmica, resistência, ignitabilidade, densidade, condutividade térmica e reação ao fogo conforme o disposto na tabelas 13, apresentada na Figura 6.

Figura 6 - Tabela de Requisitos para Isolantes EPS e XPS da NBR 16970 (ABNT, 2022)

Especificação	Requisito	Métodos de ensaio
Absorção de água	Variação do volume original para o volume após ensaio ≤5 % (valor após 24 h de ensaio)	ABNT NBR 7973
Estabilidade térmica	Resistir às tensões no valor mínimo de compressão de 0,10N/mm2, com variação de espessura menor que 5 %, após exposição do material por dois dias a uma temperatura de 70 °C	BS EN 13163 (para EPS) e BS EN 13164 (para XPS)
Ignitabilidade	Fs ≤ 150 mm em 20s,	ABNT NBR 15575-4, ABNT NBR 15575-5 ISO 11925-2
Densidade	Mínima 18 kg/m³	ABNT NBR 11752 ASTM C 578
Condutividade térmica a 24 °C	≤ 0,038 W/mk	ABNT NBR 11752 ASTM C 518
Reação ao fogo	Deve atender como mínimo a classe II A	ABNT NBR 9442

Fonte: ABNT (2022)

Assim, também, as Lãs de PET, de vidro e de rocha, que são isolantes térmico e acústicos, devem atender aos requisitos de comprimento e largura, gramatura média, absorção de umidade, resistência à tração, estabilidade dimensional, reação ao fogo e condução térmica (ABNT, 2022).

Quanto às barreiras de vapor e umidade, estas devem atender às especificações da Tabela 16 da norma, apresentada na Figura 7.

Figura 7 - Tabela de requisitos para a barreira de vapor e umidade da NBR 16970 (ABNT, 2022)

Especificação		Requisito	Métodos de ensaio
Resistência à tração [b]	Direção longitudinal	Mínimo 178 N	ASTM E 2556/ E 2556M-10
	Direção transversal	Mínimo 156 N	
Barreira impermeável ao vapor d'água		Permeância ≤ 0,1 perm [a]	-
Barreira permeável ao vapor d'água		Permeância ≥ 5 perm [a]	ASTM E 2556/ E 2556M-10
Impermeabilidade à água		Não pode haver formação de gotas de água na face oposta à face exposta à coluna de água de 55 cm de altura por um período de 5 h.	-

[a] 1perm = US perm = 57,2 ng/(s.m².Pa).
[b] requisito apenas para mantas pré-fabricadas.

Fonte: ABNT (2022)

Finalizando a primeira parte da NBR 16970 (ABNT, 2020), tem-se o capítulo que apresenta os requisitos de desempenho para cada sistema integrante onde são referenciadas outras normativas específicas as quais os mesmos se remetem. Neste capítulo a norma aborda requisitos de desempenho para os sistemas de: estrutura; vedações verticais; subsistema de piso; subsistema de cobertura; durabilidade e manutenibilidade da edificação, abordando, dentre outros, requisitos como os de resistência a impactos de corpo mole, resistência às solicitações de cargas suspensas, resistência a impactos de corpo duro, solicitações transmitidas às paredes pelas aberturas, segurança contra incêndio, estanqueidade à água, impermeabilidade e desempenho termoacústico (ABNT, 2022).

Destaca-se que a norma define que o projeto e a execução dos sistemas em *Light Steel Frame* devem considerar uma vida útil de projeto (VUP) de no mínimo 50 anos para estrutura e vedação externa e de 20 anos para vedação interna, se submetidos a manutenções preventivas e corretivas sistemáticas conforme previsão do manual de operação, uso e manutenção.

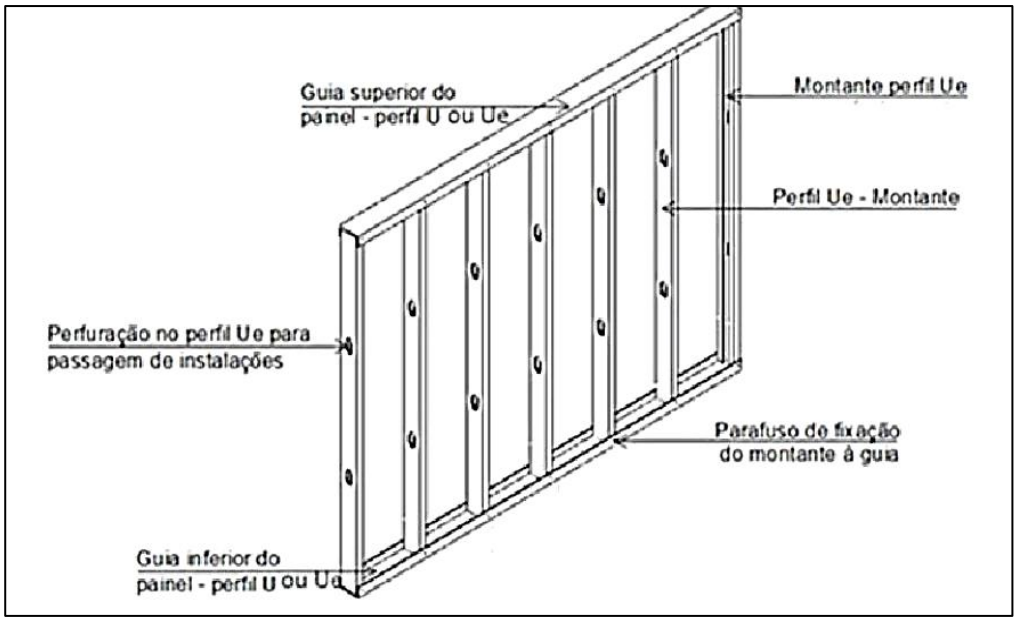
Neste ponto, a norma também aponta que para atender os requisitos de manutenibilidade da edificação, deve-se estabelecer em manual de uso, operação e manutenção, os prazos de vida útil do projeto e de suas diversas partes, bem como os procedimentos e materiais necessários para limpezas, serviços de manutenção e reparos, substituição de componentes, fixação de peças suspensas nas paredes, dentre outros.

2.2.2 Projeto Estrutural

A segunda parte da Norma Brasileira de *Light Steel Frame* - NBR 16970-2 (ABNT, 2022), define os padrões de projeto estrutural e estabelece os requisitos gerais para o dimensionamento de estruturas constituídas por perfis formados a frio. Ou seja, dos componentes estruturais do tipo perfis, elementos de fixação, contenção lateral, ancoragem, suportes, fitas metálicas, chapas de *gousset*, placas estruturais de vedação e elementos auxiliares.

Os termos e definições de norma são de painel reticulado, ou seja, um sistema estrutural plano constituído por perfis ligados entre si. Pode ser constituído pelas guias inferiores e superiores de perfil U ou Ue, montantes de perfil UE e parafusos de fixação do montante a guia conforme a Figura 8.

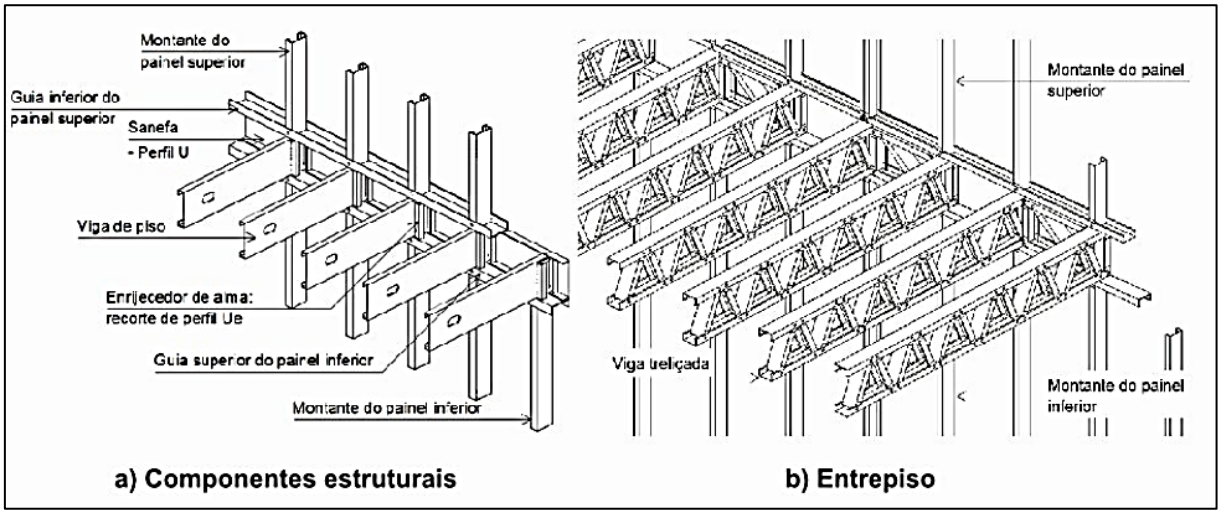
Figura 8 - Painel reticulado e seus componentes



Fonte: ABNT (2022)

O entrepiso é o conjunto de construção compreendido entre o nível do forro e o piso do pavimento imediatamente superior e pode ser composto pela Sanefa de perfil U, as vigas de piso, enrijecedor de alma de perfil Ue e também pelas vigas treliçadas, conforme apresentado na Figura 9.

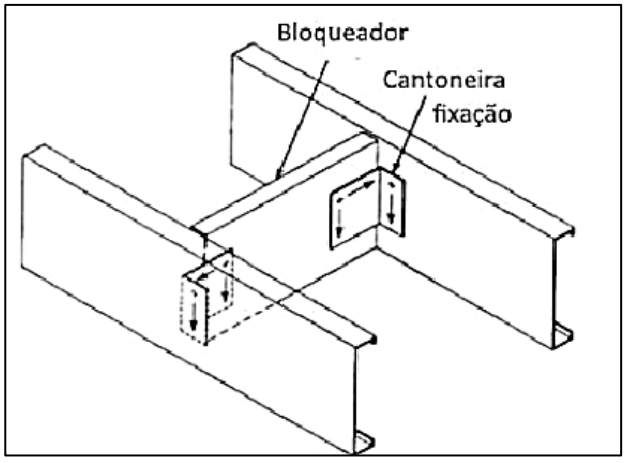
Figura 9 - Entrepiso e seus componentes estruturais



Fonte: ABNT (2022)

Bloqueadores são perfis de aço normalmente utilizados na horizontal para a contenção lateral dos componentes estruturais. Eles são fixados por meio de cantoneiras de fixação, conforme mostra a Figura 10.

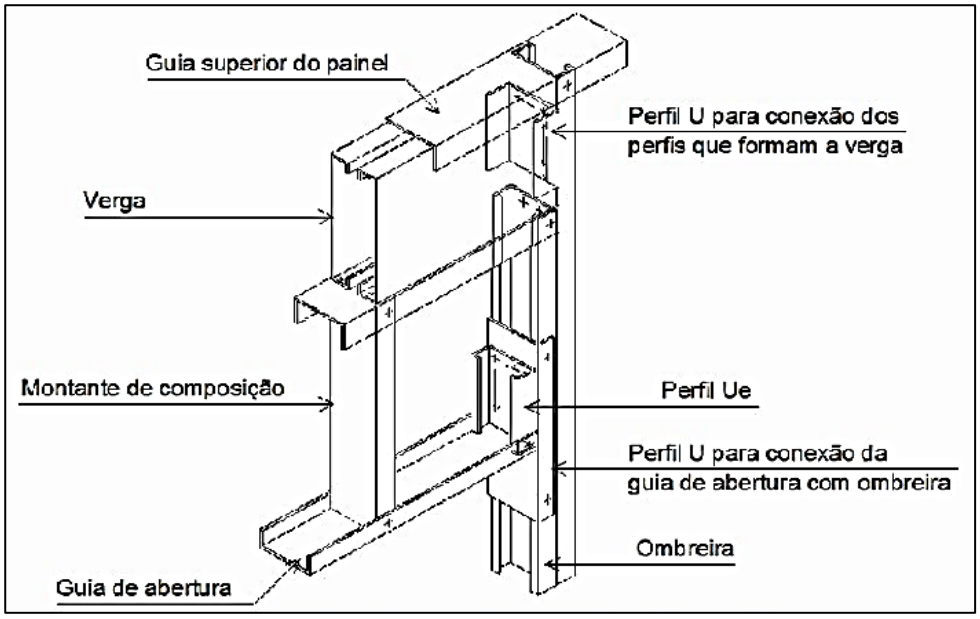
Figura 10 - Elemento bloqueador



Fonte: ABNT (2022)

As vergas, ou seja, vigas para transição de forças situadas ou posicionadas na parte superior de aberturas deve ser projetada conforme especificações e detalhamento juntamente com os elementos das barras de composição conforme apresentados na Figura 11.

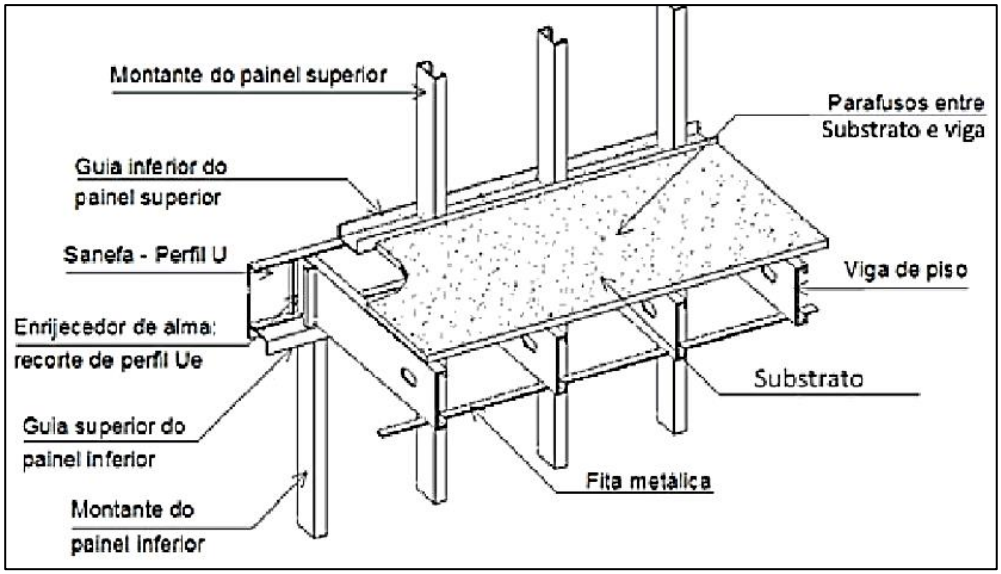
Figura 11 - Detalhes de verga e das barras de composição conforme a NBR 16790



Fonte: Associação Brasileira de Normas Técnicas (2022)

Fitas são elementos utilizados no sistema de piso, juntamente com outros para conformar um sistema de laje seca conforme visto na Figura 12.

Figura 12 - Detalhes da laje seca conforme a NBR 16790

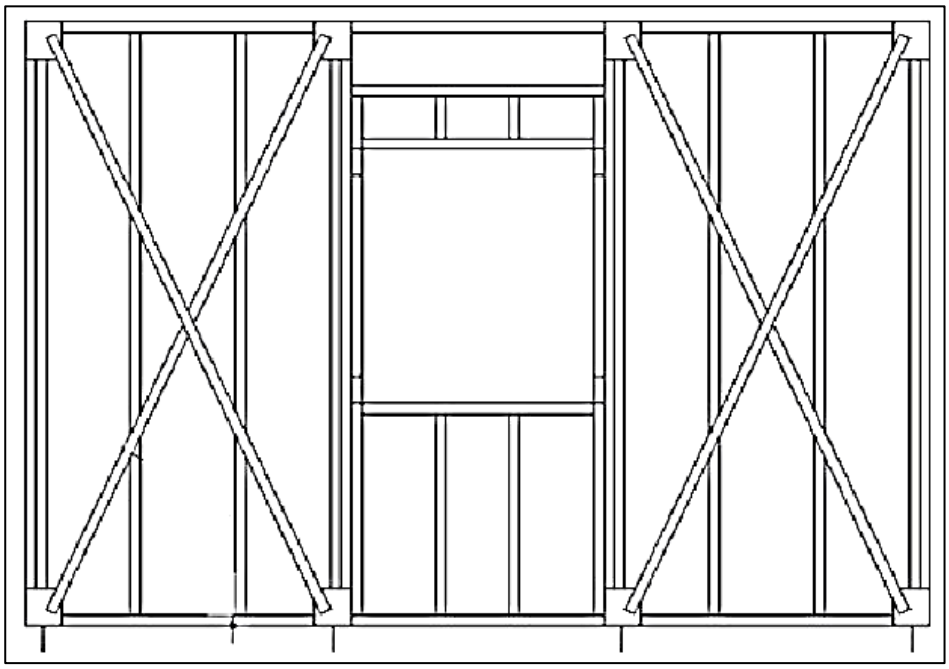


Fonte: ABNT (2022)

Já a laje do tipo úmida é aquela dos quais são utilizados elementos compostos do sistema construtivo LSF sobre a qual são executados outros tipos de piso como, por exemplo, concreto armado.

As contenções laterais e horizontais podem ser aplicadas para restringir ou reduzir os deslocamentos e as distorções dos painéis. Estas contenções podem ser executadas com elementos do tipo treliça ou fita, um exemplo desta aplicação pode ser visto na Figura 13.

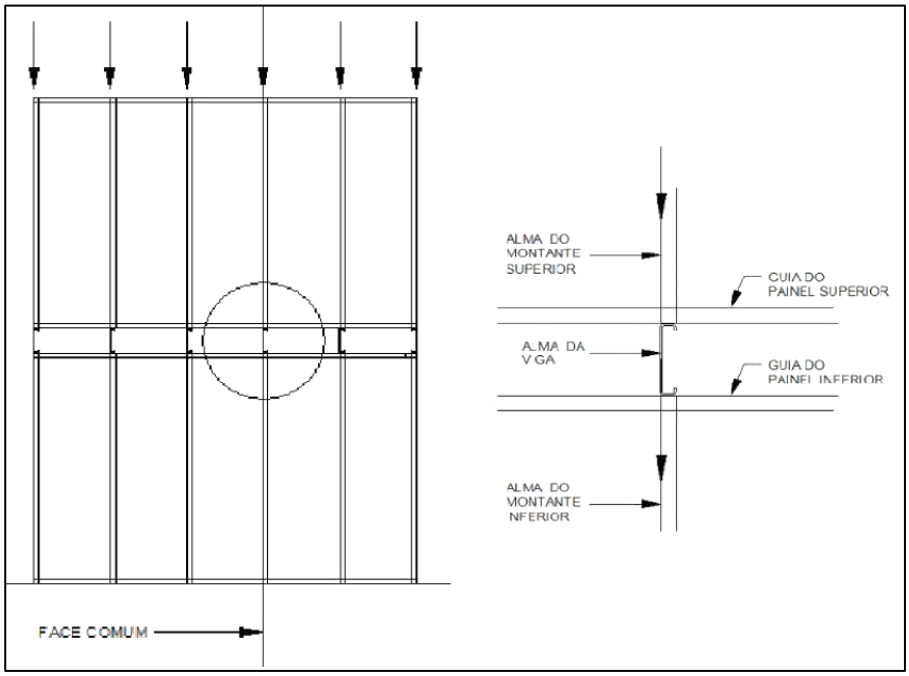
Figura 13 - Detalhes de contenção lateral de painel com fita conforme a NBR 16790



Fonte: ABNT (2022)

Uma importante característica dos princípios e generalidades de projeto em LSF definido em norma é que os montantes dos painéis dos pavimentos devem estar alinhados entre si. Estes também devem estar alinhados com as vigas de entrepisos e com a estrutura do telhado. A esta característica utiliza-se o termo *in-line framing* e pode ser entendida através da Figura 14. Deve-se atentar também para o limite de excentricidade tolerada para estes alinhamentos que não pode ser superior a 20mm caso o eixo do enrijecedor da alma seja interno a viga; ou de 3 mm caso o enrijecedor de alma seja externo à viga (ABNT, 2022).

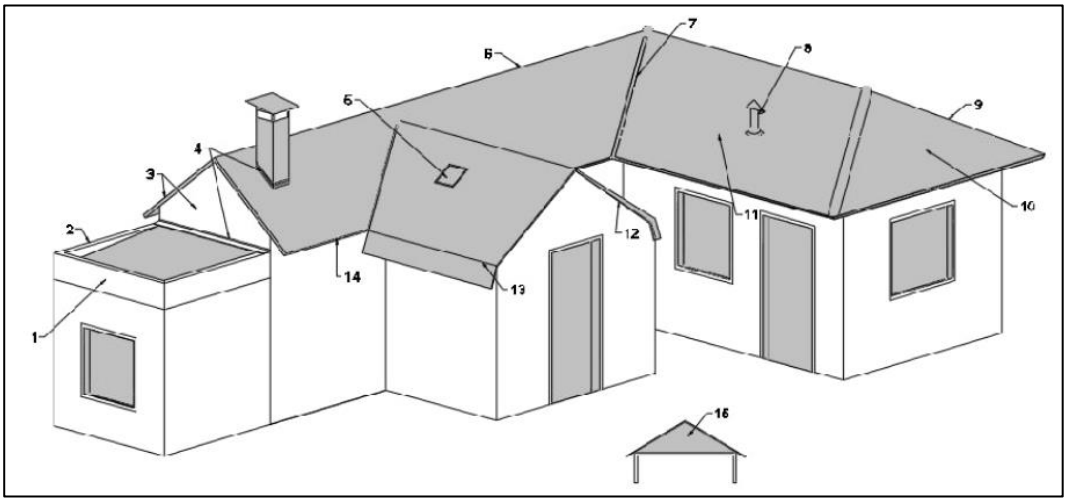
Figura 14 - In-line framing



Fonte: ABNT (2022)

As estruturas de cobertura, segundo a norma, são definidas como mostrado na **Figura 15**.

Figura 15 - Desenho de formas das coberturas

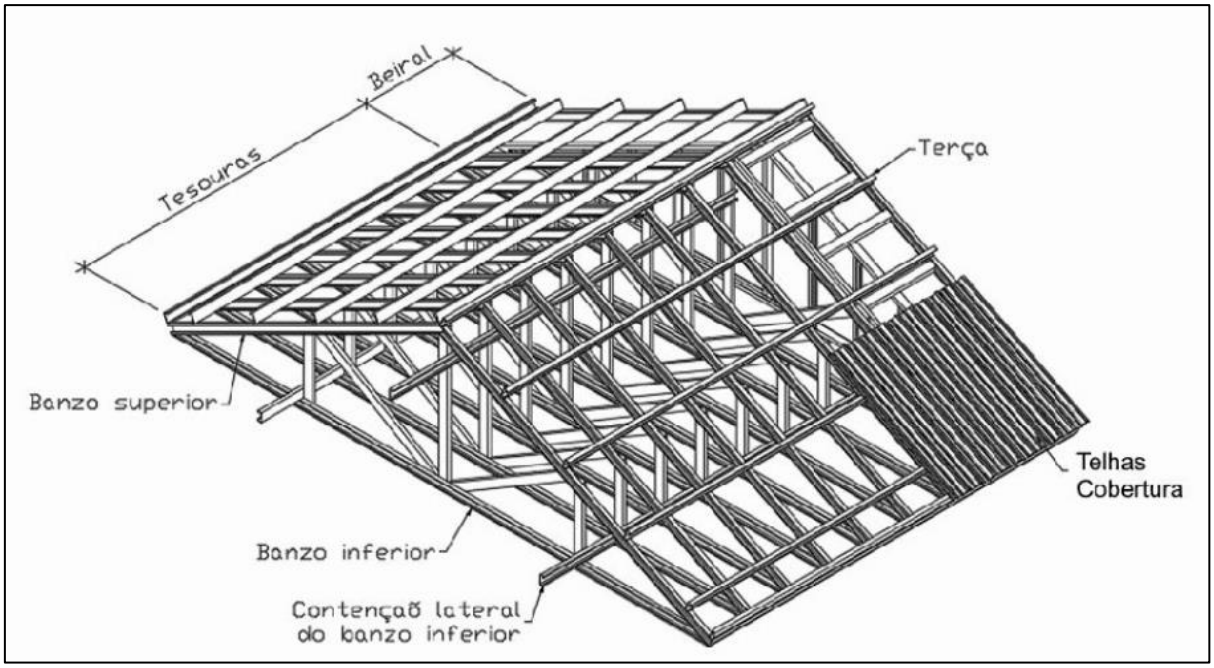


Legenda: (1) platibanda (2) cobre-muro (3) empena (4) rufo (5) claraboia (6) cumeeira (7) rincão/água furada (8) ventilação de esgoto (9) espigão (10) tacanica (11) água-mestra (12) tabeira (13) quebra (14) beiral (15) oitão.

Fonte: ABNT (2022)

Bem como do esquema da estrutura dos elementos estruturais da cobertura conforme a Figura 16.

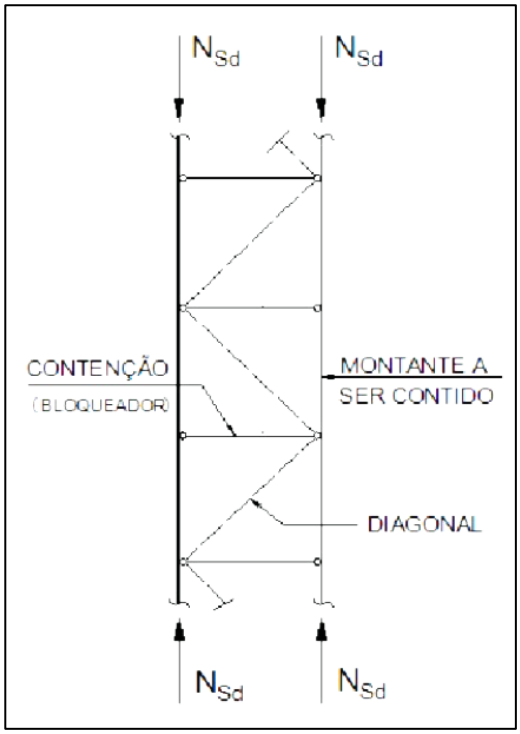
Figura 16 - Esquema da estrutura da cobertura



Fonte: Associação Brasileira de Normas Técnicas (2022)

Mais adiante a norma apresenta um conjunto de especificações, cálculos estruturais e desenhos de fabricação, da montagem e dos componentes que devem ser detalhados em atendimento aos atributos de construtibilidade, segurança e de utilização. Assim, devem atender todas as combinações de ações possíveis e aos estados limites último e de serviços estabelecidos pela norma NBR 14762 (ABNT, 2001b). A Figura 17 mostra um exemplo de montante sendo contido pelos bloqueadores (ABNT, 2022).

Figura 17 - Esquema de contenção em colunas de barras comprimidas



Fonte: ABNT (2022)

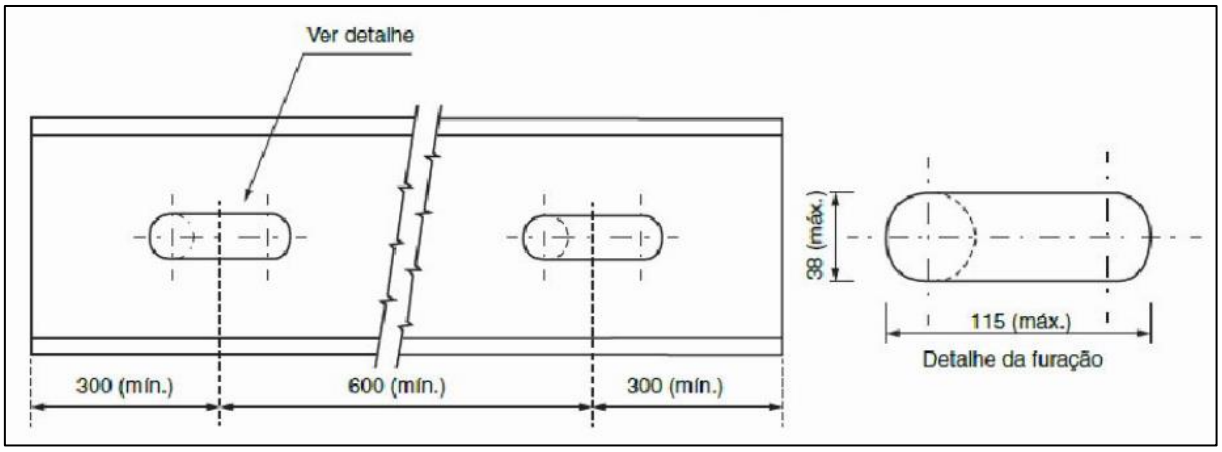
Também, são discorridos sobre os deslocamentos máximos verticais e horizontais, permitidos pela norma, para a verificação de estado-limite de serviço para deslocamentos excessivos. É estabelecido que os deslocamentos horizontais devem ser analisados observando-se os métodos de segunda ordem e a comparação com os valores da NBR 8800 (ABNT, 2008). Bem como das vibrações e ressonância em pisos, onde são introduzidas equações a fim de avaliar a resposta provocada pelos pisos, além de critérios de aceitabilidade.

2.2.3 Interface entre Sistemas

A terceira parte da NBR 16970 (ABNT, 2022) trata sobre as interfaces entre sistemas, ou seja, as determinações e os detalhes construtivos para interfaces entre os sistemas que compõem a edificação. Assim, devem ser consideradas as interfaces entre paredes e pisos, externos e internos, entre paredes e esquadrias, entre paredes ou pisos e instalações. Um exemplo sugerido em norma é de alguns cuidados na instalação de esquadrias como o envelopamento dos vãos com materiais específicos como prevenção de fissuras e infiltrações (ABNT, 2022).

Outro exemplo de interface entre sistemas tratados pela norma, são quanto às instalações prediais elétricas, hidráulicas, de gás, de SPDA, de ar-condicionado, dentre outros, recomendando-se a utilização de *shafts*, ou se necessário, da previsão de furação para passagens de instalações de acordo com a NBR 15253 (ABNT, 2014), a qual admite a execução de aberturas sem reforços nos perfis, desde que sejam devidamente consideradas no dimensionamento e que o eixo maior coincida com o eixo longitudinal da alma do perfil, conforme a Figura 18.

Figura 18 - Detalhe de espaçamento e furações nos perfis



Fonte: ABNT (2022)

Já, a respeito da fixação de objetos em paredes, estes devem ser fixados diretamente nas chapas, nos perfis de aço ou em reforços aplicados internamente às paredes e aos revestimentos. E, nas áreas molhadas e molháveis, além da impermeabilização de piso, devem-se atender aos requisitos da NBR 15758 (ABNT, 2009).

2.3 Desenvolvimento de Famílias no *AUTODESK REVIT*

Famílias são um grupo de elementos com um conjunto de parâmetros, ou propriedades em comum, e suas representações gráficas. Já as variações destas famílias são chamadas de Tipos. Pode-se considerar estas famílias como blocos construtivos dentro do ecossistema do *REVIT*, porém suas funções vão muito além disto.

2.3.1 Tipos de Famílias

O *REVIT* possui algumas famílias denominadas Famílias de Sistema, as quais são carregadas por padrão em todos os projetos e não podem ser carregadas separadamente, como por exemplo, as famílias de níveis, paredes, pisos e massas. Outro tipo de família são as denominadas “Carregáveis”. As Famílias Carregáveis podem ser introduzidas aos projetos e podem ser desenvolvidas pelos utilizadores da ferramenta, por fabricantes e por empresas especializadas.

Alguns exemplos de famílias do *REVIT* são: Balaústres (corrimão e guarda-corpo), *Casework* (equipamentos ou móveis), *Doors* (portas), Electrical Fixtures (símbolos para projetos elétricos), *Entourage* (blocos de RPC’s - ou seja, de personagens, utilizados para humanizar maquetes eletrônicas), *Furniture* (moveis), *Light Fixtures* (luminárias), *Mass* (sólidos), *Mechanical Equipament* (ar condicionado), *Planting* (plantas), *Profiles* (rodapés, sancas, molduras, calhas, dentre outras), *Site* (postes, vagas de estacionamento, mobiliário urbano), *Speciality Equipment* (elevadores, escritórios, eletrodomésticos, eletrônicos), *Structural* (vigas, pilares, junções, amarrações e travamentos) e *Tibleblocks* (pranchas) (*AUTODESK*, 2022).

2.3.2 Modelos e Hospedeiros

Os Arquivos de Modelos, ou *Templates,* são a base para a criação de famílias ou projetos no *REVIT*. Ou seja, tudo que é criado, alterado ou modificado, é feito a partir de algo preexistente. Por exemplo, na criação de uma família de porta, deve-se partir do *template* de porta (*door*) e deve-se respeitar os hospedeiros desta classe de família. Isso é necessário pois o *REVIT* agrupa as famílias em categorias com características similares, como por exemplo, o grupo das luminárias de teto que só podem ser fixadas em foros. Ou seja, nesse caso, o forro pode ser hospedeiro (*host*) do tipo luminária de teto, da mesma forma que portas verticais podem ser hospedadas em paredes e muros (*AUTODESK*, 2022).

2.3.3 Ferramentas de Formas

As ferramentas de forma, permitem a criação de formas geométricas tridimensionais a partir de formas bidimensionais para construir a representação

gráfica das famílias. As ferramentas básicas de forma disponíveis no *REVIT* são: Extrusão, Mescla, Revólver, Varredura e Mescla com Varredura.

A Extrusão permite converter um desenho bidimensional em uma geometria tridimensional e perpendicular ao desenho, ou seja, o formato da base é mantido e a forma *"nasce"* a partir dela. A Mescla permite gerar uma extrusão entre dois desenhos, onde a extrusão vai se modificando entre a primeira forma até adotar a forma da outra.

Revolver é uma ferramenta que permite criar uma extrusão ao redor de um eixo, criando uma forma tridimensional cilíndrica, ou seja, uma forma de revolução. A Varredura permite criar uma extrusão linear que percorre um caminho, como por exemplo, uma linha. E a Mescla com Varredura faz a mescla de dois desenhos bidimensionais através de uma extrusão que percorre uma linha entre estas duas formas.

Já as Formas de Vazio, permitem a extrusão de formas tridimensionais de cortes ou furos sobre outros elementos e possuem suas próprias versões de todas as ferramentas de extrusão já citadas (AUTODESK, 2022).

2.3.4 Plano de Referência e Plano de Trabalho

Planos de Referência são elementos planos em um espaço tridimensional que pode servir como linhas de esboço ou como ferramenta de controle de geometrias na criação de famílias. Neste sentido, os níveis são planos de referência.

Os Planos de Referência permitem criar restrições às geometrias do projeto, de forma que ao movimentar o plano de referência, a geometria também se move. Isso é feito com a utilização da ferramenta "alinhar" e da opção de criação de restrição (cadeado).

E os Planos de Trabalho são aqueles sobre os quais se está modificando as linhas de desenho. Diversos planos de trabalho são criados automaticamente pelo *REVIT* para cada vista trabalhada e também é possível criar planos de trabalho personalizados com referência a partir de qualquer linha do desenho (*AUTODESK*, 2022).

2.3.5 Configurações Iniciais para Criação de Famílias

Ao iniciar um projeto de criação de família, deve-se determinar qual a

categoria e parâmetros de família que deverão ser selecionados. Existem propriedades predefinidas de categorias sobre os componentes que serão produzidos. Assim, para selecionar a configuração dos parâmetros e seus efeitos de aplicação, deve-se proceder o acesso à ferramenta “Categoria e parâmetros de família” da aba “criar”. O Quadro 1 apresenta alguns parâmetros de famílias e seus efeitos de aplicação.

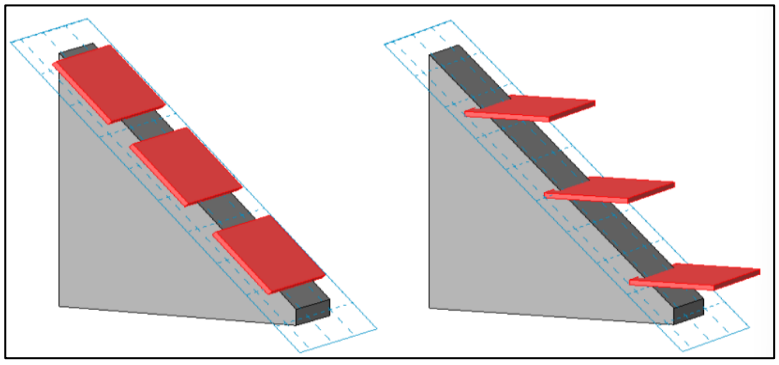
Quadro 1 - Exemplos de parâmetros e seus efeitos de aplicação

Parâmetro	**Efeito de aplicação**
Hospedeiro	Exibe o hospedeiro para uma família com base em hospedeiro, tal como uma parede
Com base no plano de trabalho	A família é hospedada pelo plano de trabalho ativo
Sempre na vertical	A família sempre aparece na vertical em 90 graus, mesmo se estiver em um hospedeiro inclinado.
Cortar com vazio quando carregados	Os vazios criados na família irão cortar através dos sólidos. As categorias que podem ser cortadas pelo vazio são: forros, pisos modelos genéricos, telhados, pilares estruturais, fundações estruturais, framing estrutural e paredes
Tipo de peça	Fornece uma classificação adicional para uma categoria de família e determina o comportamento da família no modelo

Fonte: Adaptado de AUTODESK (2022)

Por exemplo, é possível tornar uma família em “com base no plano de trabalho” e em “sempre na vertical” ou “não sempre na vertical”. Exemplos de ambos os comportamentos são demonstrados abaixo na Figura 19.

Figura 19 - Exemplo de aplicações



Fonte: ABNT (2022)

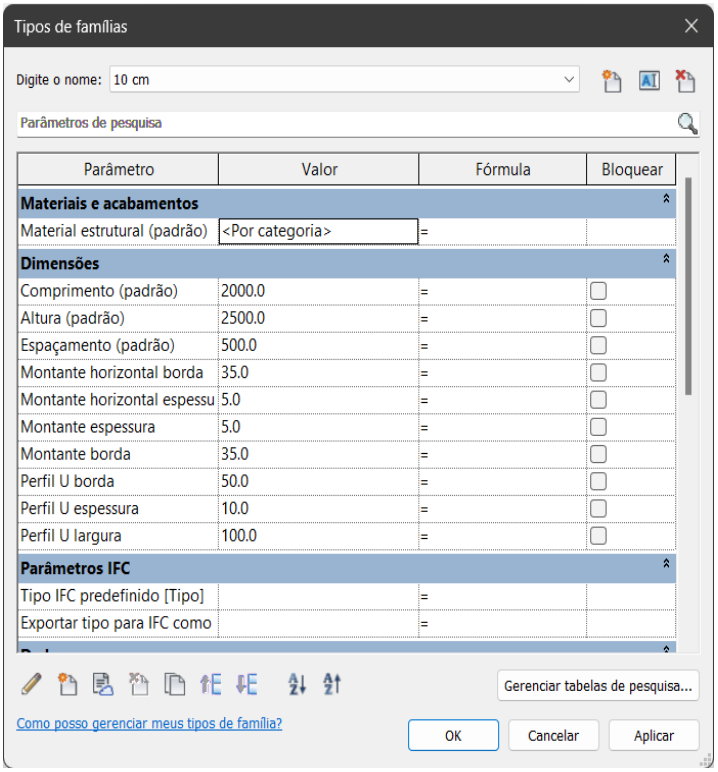
Outra configuração inicial diz respeito à determinação das unidades de projeto da família que pode ser acessada através da ferramenta “Unidades de Projeto” da aba “Gerenciar”. Ao acessá-la é possível definir os formatos, unidades, números de casas decimais, a aplicação de símbolos e as configurações de supressão de zeros, para cada disciplina de projeto.

2.3.6 Parâmetros

Parâmetros são as características configuráveis dos objetos, ou famílias, como por exemplo, a instância de uma família de porta, pode ter diversos parâmetros configuráveis, como a largura, altura, cor, material, direção, dentre outras, podendo também controlar o comportamento dos objetos.

Para acessar a lista de parâmetros criados, bem como do seu gerenciamento, deve-se utilizar a ferramenta “Tipos de família” da aba “Criar”. Nela, os parâmetros são listados em grupos fixos, os quais podem ser criados, alterados ou excluídos conforme demonstrado na Figura 20.

Figura 20 - Ferramenta tipos de famílias



Fonte: Autoria Própria (2022)

Esses parâmetros, ou propriedades, podem ser definidos para cada instância das famílias, não afetando as outras instâncias, ou, podem ser criadas Famílias-Tipo, que são subgrupos da Família das quais as mudanças nas propriedades são propagadas para as suas instâncias (AUTODESK, 2022).

2.3.7 Aplicação de Fórmulas em Parâmetros

As fórmulas são formas de aplicar alterações aos parâmetros com base em cálculos e decisões lógicas, ou seja, através de declarações condicionais de controle. Elas são definidas na coluna “Fórmula” da ferramenta “Tipos de Famílias” apresentada na Figura 20, e suporta sintaxe de operações aritméticas, funções trigonométricas e operações lógicas conforme os exemplos apresentados no Quadro 2.

Quadro 2 - Exemplos de Operadores, funções e descrições de fórmulas

Operador/Função	**Descrição**
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
^	Exponenciação
log	Logaritmo
ln	Logaritmo natural
sqrt	Raiz quadrada
sen	Seno
cos	Cosseno
tan	Tangente
round(x)	retorna o valor “x” arredondado
IF (condição, verdadeira, falso)	se a condição for verdadeira, aplica a fórmula denominada “verdadeira”, caso contrário, a fórmula definida em “falso”

Fonte: Adaptado de *AUTODESK* (2022)

2.4 API do *REVIT* e o Desenvolvimento de *Plug-ins*

Mesmo sendo uma poderosa ferramenta, o *REVIT* possui algumas limitações no escopo do que pode ser desenvolvido e que está implementado através de suas interfaces. Dessa forma, para garantir a capacidade de ampliação de suas funções de forma a abarcar todas as necessidades dos usuários, o *REVIT* disponibiliza algumas formas de interação com seu sistema através de linguagens de programação. Isso permite o desenvolvimento de plugins, ou seja, de complementos a sua funcionalidade.

A *REVIT Application Programming Interface* - *REVIT* API, ou Interface de Programação de Aplicação, em tradução direta, é a interface que possibilita a conexão de diferentes aplicativos ao *REVIT*. Ela funciona como um mensageiro que recebe as requisições aos mecanismos internos da aplicação e entrega a informação ou resposta. Ela também é responsável pelos mecanismos de controle de permissões e de manutenção de integridade dos dados.

A API do *REVIT* comina a modelagem paramétrica BIM e a programação orientada a objetos, permitindo a interação com os elementos do *REVIT* através de métodos e manipulação de todos os parâmetros ao passo que também permite, a depender da linguagem de programação utilizada, o acesso às mais diversas bibliotecas de funções úteis às operações (YANG; GRUSSENMEYER, 2018).

No BIM, cada um dos elementos inseridos no projeto correspondem a modelagem dos objetos reais e dos sistemas construtivos em que todas as representações físicas, de materiais, de resistência, dentre outros, podem ser relacionados e interferir nos outros elementos através de regras definidas por meio de parâmetros controláveis e de regras de relacionamento. Ou seja, o comportamento de um objeto do modelo porta pode se comportar de maneiras distintas a depender do tipo de objeto que o hospeda, ou dos parâmetros gerais do projeto.

Todas as informações dos elementos de projeto são registradas no banco de dados do modelo. São estas informações que podem ser acessadas e manipuladas por meio de linguagens de programação como *Visual Basic*, C++, C#, *JavaScript* e *Python*. Assim, pode-se inserir elementos, criar e alterar parâmetros, checar inconsistências, criar arranjos, alterar geometrias, dentre outros (SENA, 2019).

Um programa de computador é normalmente criado através de um código de programação que traduz fielmente a sequência lógica de instruções necessárias para

cumprir a execução do algoritmo que se deseja implementar. Na programação orientada a objetos é feito um paralelo entre os objetos do mundo real e seus relacionamentos, ou seja, os objetos são modelados para refletirem suas propriedades, funções e formas de interação através de entradas e saídas de dados entre estes elementos. Logo, este paradigma de programação é o mais indicado para interação com modelos objetificados do BIM.

Cada linguagem de programação possui aspectos que as distinguem e que podem ser um fator a se considerar pois podem impactar diretamente na forma, dificuldade e limitações técnicas, principalmente de acesso à documentação e bibliotecas. Porém, quase todas as linguagens possuem a característica comum de tratar um conjunto de dados de entrada, através de cálculos matemáticos, expressões lógicas proposicionais e por laços de repetição e execução condicional, para gerar uma saída de dados processada.

A linguagem de programação utilizada neste trabalho foi a *Python*, uma linguagem de alto nível, orientada a objetos, gratuita e disponível em várias plataformas e sistemas operacionais. Além disso, está disponível, de forma gratuita, uma vasta gama de conteúdos de referência, em especial, sobre as rotinas de manipulação da *REVIT* API, pois é a linguagem incorporada na outra ferramenta interna de automação do Revit, o *Dynamo*. Também, por ser uma das linguagens com mais ferramentas e bibliotecas disponíveis para a implementação de mecanismos de inteligência artificial, esta permite a ampliação dos trabalhos futuros para o desenvolvimento de soluções que incorporem aprendizado de máquina SENA (2019).

Cada aspecto da API do *REVIT*, bem como dos objetos, propriedades e métodos disponíveis para tratamento está disponível através do site de suporte da *AUTODESK*, de forma muito limitada, porém é possível ter acesso a informações um pouco mais detalhadas através do site *REVIT API DOCS* (*REVIT*APIDOCS, 2022), que se trata de uma documentação não oficial e gratuita.

3 METODOLOGIA

O presente trabalho foi realizado, inicialmente, através de uma pesquisa bibliográfica acerca do método construtivo *Light Steel Frame* e sobre o desenvolvimento de famílias para o *software AUTODESK REVIT*. Este estudo foi realizado por meio de uma revisão qualitativa de literatura, realizada pela análise de livros, dissertações, normas técnicas e artigos científicos, selecionados por meio de leitura exploratória em busca da sua essência útil e relacionada aos assuntos abordados.

Seguindo-se a modelagem, projeto, implementação e teste das famílias e *plug-ins*. Estes últimos passos serão executados utilizando-se o próprio *software AUTODESK REVIT*, onde também foi desenvolvido um protótipo para avaliar a usabilidade destas famílias e ferramentas.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

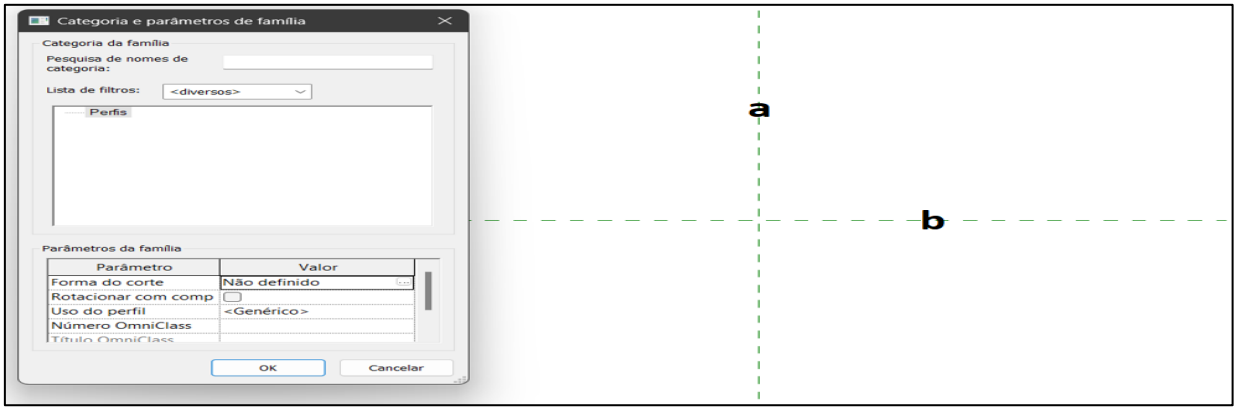
A modelagem paramétrica das famílias e a geração de protótipos de solução foi separada em três etapas, iniciando-se com a criação dos componentes básicos na forma de famílias, ou seja, dos montantes, guias, bloqueadores e conexões. Posteriormente, estes foram agrupados para formar os subsistemas de painéis reticulados da estrutura em paredes arquitetônicas hospedeiras. Este agrupamento deu-se pela modelagem e implementação de ferramentas na forma de *Plug-ins*; e finalmente com a modelagem e implementação de ferramentas de gerenciamento e documentação de projeto.

4.1 Modelagem das Famílias

A modelagem das componentes no *REVIT* seguiu um processo de estruturação em etapas sequenciais, e necessárias, para definir o comportamento básico destes componentes. Esse processo teve início com a escolha do *template* e categoria de família, ou seja, do carregamento de um modelo básico de família genérica e com dados de parâmetros pré-estabelecidos para cada categoria de componente que se desejou criar.

O *template* escolhido para os modelos de perfis metálicos foi o “Perfil métrico”. Este template apresenta dois planos de referência centrais ao elemento e a categoria de famílias “Perfis” pré-estabelecida, como mostrado na Figura 22.

Figura 21 - Planos de Referência de Origem

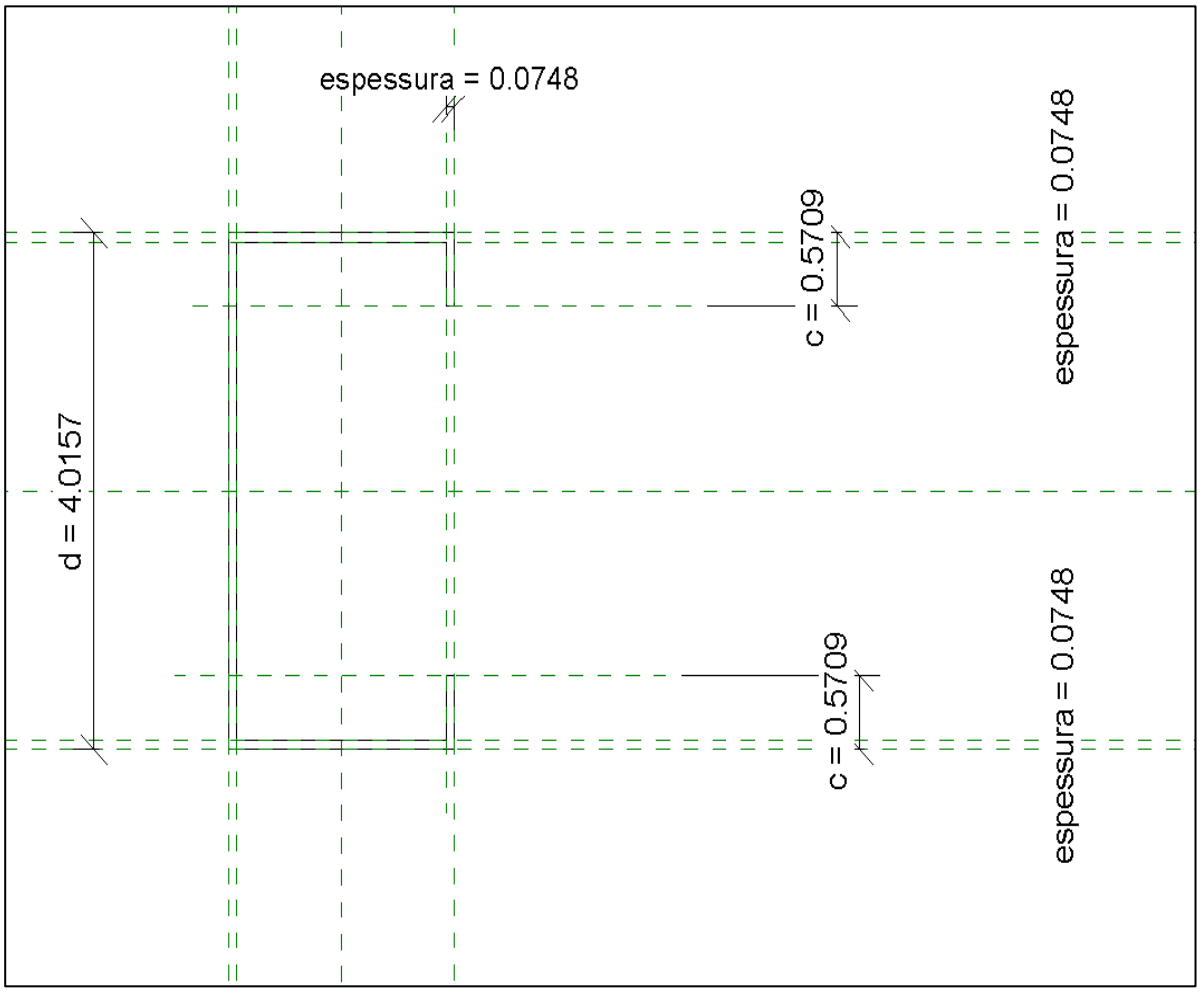


(a) Plano de referência central vertical; (b) Plano de referência central horizontal

Fonte: Autoria Própria (2022)

A modelagem da geometria do componente ocorreu através da definição de diversos planos de referências que foram então ancorados como restrições à e relações entre parâmetros de cotas e as linhas de desenho do modelo. A Figura 23 mostra estes planos para o perfil “C” enrijecido – “Ce”.

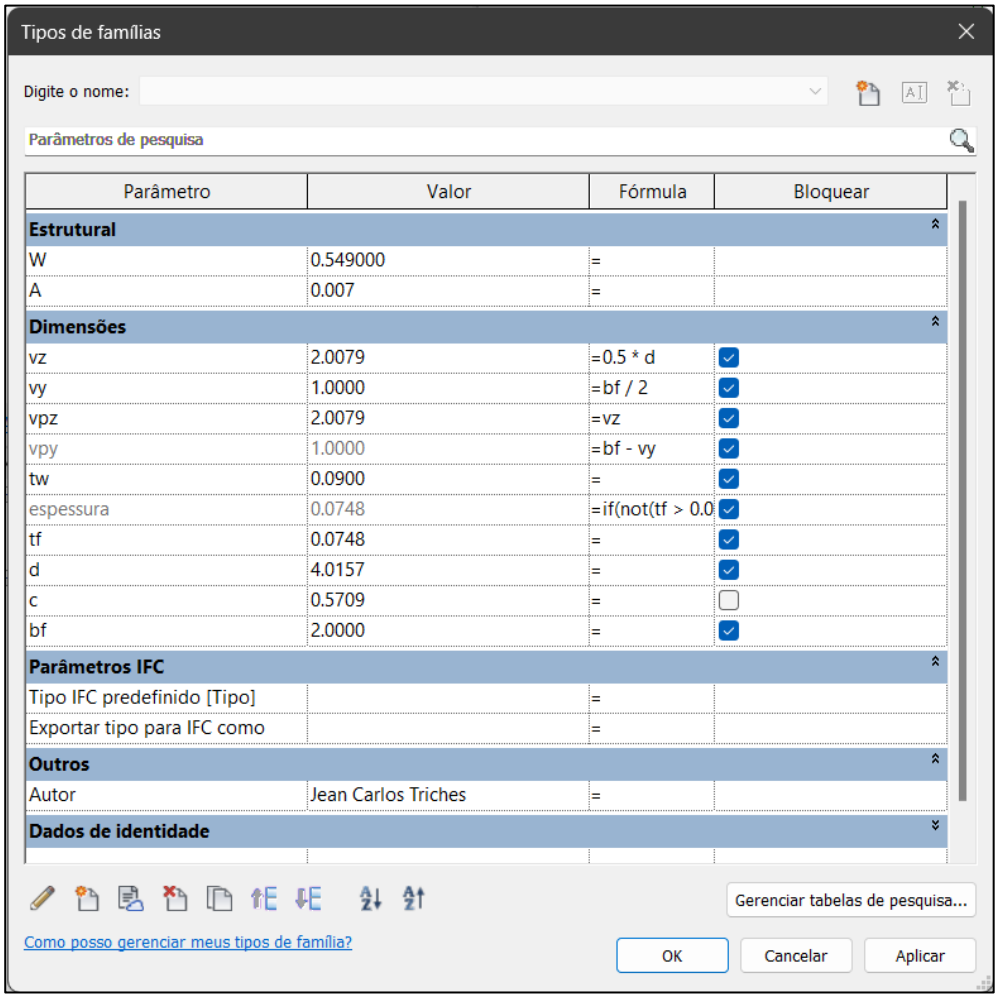
Figura 22 - Planos de Referência na Modelagem de um Perfil



Fonte: Autoria Própria (2022)

Os parâmetros relacionados a cada cota do perfil foram definidos através da ferramenta “Tipos de Famílias” onde foi possível também definir fórmulas, conforme a Figura 24 mostra.

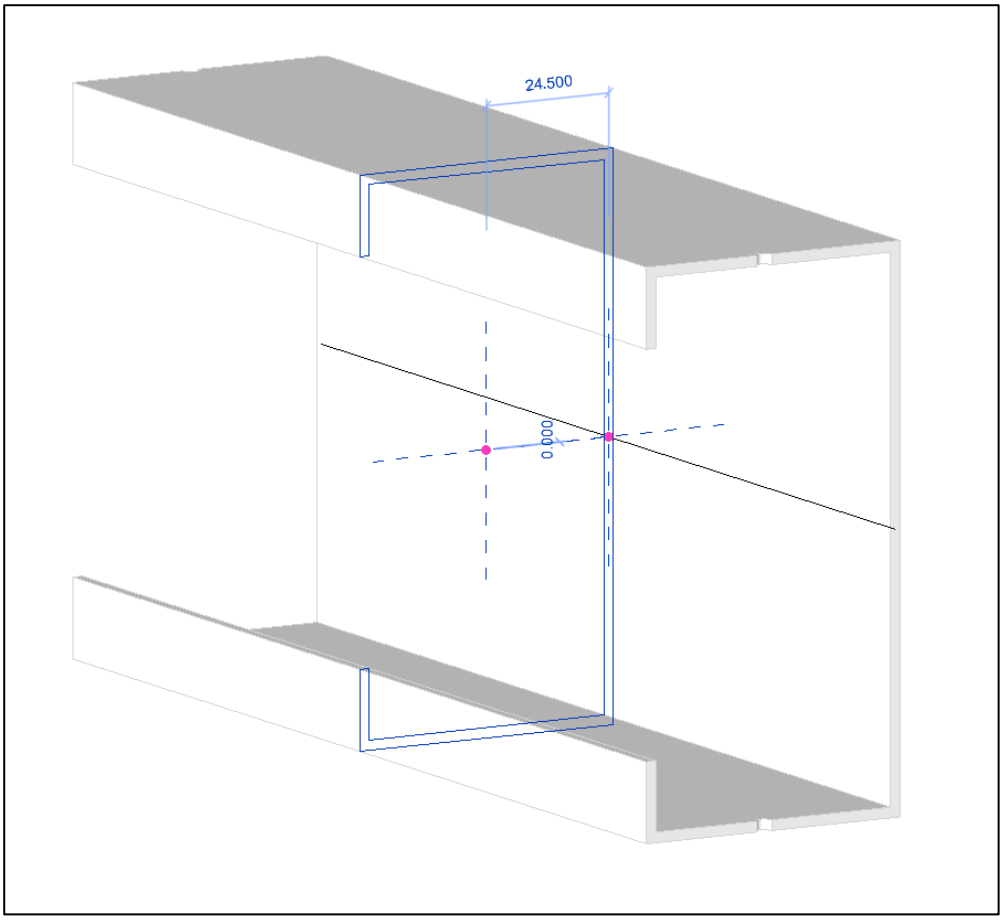
Figura 23 - Tipos e Formulas de Famílias



Fonte: Autoria Própria (2022)

Esse procedimento foi realizado tanto para a criação das famílias que definem o desenho dos perfis metálicos como componentes básicos do sistema, ou seja, dos montantes, guias, fixadores e bloqueadores. Para a modelagem das famílias do tipo montante e guias, seguiu-se da definição de uma massa formada pela varredura do perfil através de um caminho definido entre dois outros planos de referência cuja distância entre eles foi parametrizada como o comprimento da peça. A Figura 25 mostra uma visão tridimensional da família de montante “Ce”. Nela é possível verificar o posicionamento da forma de perfil, bem como da linha de varredura que este perfil percorre para formar a peça. Para estes componentes, também foram definidos os parâmetros básicos da categoria de famílias “Quadro Estrutural”.

Figura 24 - Definição do Perfil de Varredura



Fonte: Autoria Própria (2022)

Assim, como foram criadas famílias parametrizadas para estes componentes estruturais, pode-se definir quaisquer valores de tamanhos e espessura para as propriedades da sessão.

4.2 Modelagem dos *Plug-in*s

Já para a criação dos componentes dos subsistemas compostos por painéis reticulados, foi necessário a criação de um *plug-in*, ou seja, de uma extensão desenvolvida através da linguagem de programação *Python*, responsável por transformar estas famílias parametrizadas em um subsistema de painéis estruturais hospedados em paredes arquitetônicas do projeto.

Tal necessidade advém das limitações que as ferramentas nativas de criação de famílias paramétricas do *REVIT* apresentam para a inserção e controle de hospedagem de componentes como janelas, portas, dutos e aberturas em famílias

diversas da de parede arquitetônica. Assim optou-se pelo desenvolvimento de uma ferramenta capaz de: através de um modelo de parede arquitetônico, criar os painéis estruturais construtivos, considerando a interferência com os componentes hospedados na parede.

A ferramenta principal do *plug-in*, ou seja, a de criação dos painéis, foi desenvolvida através do acesso aos objetos do documento, seus atributos e métodos compartilhados pela API do *REVIT*. Passa-se então a descrever as principais características do algoritmo.

A API disponibiliza a interface com os objetos principais do controle do documento do projeto ativo no momento da execução do *plug-in* e também um objeto de controle da aplicação. Estes objetos foram acessados com os nomes “doc” e “app”, respectivamente, conforme mostrado na Figura 26.

Figura 25 - Código de acesso aos objetos

```
12    from Autodesk.Revit.DB import *
13    from Autodesk.Revit.DB.Structure import *
14    doc   = __revit__.ActiveUIDocument.Document
15    uidoc = __revit__.ActiveUIDocument
16    app   = __revit__.Application
```

Fonte: Autoria Própria (2022)

Uma das formas usadas para acessar os parâmetros dos objetos, ou elementos, que compõem o projeto foi através do método “*LookupParameter*”, como mostrado na implementação da função da Figura 27.

Figura 26 - Código de função de captura de parâmetros

```
122 def get_parameter_value_by_name(element, parameterName):
123         return element.LookupParameter(parameterName).AsValueString()
```

Fonte: Autoria Própria (2022)

Este método tenta encontrar um parâmetro nos elementos cujo nome corresponde ao parâmetro “*parameterName*”. Os resultados possíveis incluem: se um único parâmetro correspondente é encontrado, ele será retornado; se nenhum parâmetro correspondente for encontrado, retorna uma referência nula; e se existirem vários parâmetros correspondentes, retornará o primeiro encontrado.

É possível haver várias correspondências de parâmetros com o mesmo nome, pois os parâmetros compartilhados de famílias ou parâmetros de projeto podem ser vinculados a uma categoria de elemento mesmo se já houver um parâmetro interno com o mesmo nome. Isso é algo gerador de bastante confusão no desenvolvimento dos algoritmos.

Passou-se ao desenvolvimento do algoritmo responsável por identificar as paredes selecionadas (linhas 133 a 135), de recuperar os seus parâmetros necessários (linhas 136, 137, 138, 141 e 147), bem como por calcular algumas de suas características e de criar as instâncias de estrutura na lista de estruturas (linha 143), conforme mostrado no código da Figura 28. A definição completa destas estruturas de dados pode ser verificada no Apêndice 1.

Figura 27 - Código exemplo de captura e cálculo de propriedades

```
132  if __name__ == "__main__":
133      selection = [doc.GetElement(x) for x in
134          uidoc.Selection.GetElementIds()]
135      for wall in selection:
136          startPoint = wall.Location.Curve.GetEndPoint(0)
137          endPoint = wall.Location.Curve.GetEndPoint(1)
138          level_0 = uidoc.Document.GetElement(wall.LevelId)
139          comment = 'p'+str(randint(0,10000))
140          comment_analitico = 'A'+str(randint(0,10000))
141          wallWidth = cmtoinch(float(get_parameter_value_by_name(wall,
142              'Altura desconectada')))
143          estrutura = Struct(startPoint , endPoint, level_0, wallWidth,
144              comment)
145          analitico = Struct(startPoint , endPoint, level_0, wallWidth,
146              comment_analitico)
147          print('angulo estrutura ', estrutura.angle)
148          print ('wallwidth ', wallWidth)
149
150          #inicializando a trasação
151          t = Transaction(doc, 'column')
152          t.Start()
```

Fonte: Autoria Própria (2022)

Aqui também é definido uma propriedade de comentário para a parede arquitetônica selecionada (linhas 139, 143 e 144), a qual irá carregar uma marca de código para distinguir todas as peças estruturais que serão criadas relacionadas a ela.

Outro aspecto importante da metodologia foi a definição de uma estrutura de

controle de Transação, ou seja, da execução de todos os seguintes códigos de manipulação do projeto, de forma que no caso da ocorrência de algum erro de execução capaz de causar qualquer inconsistência, ou até mesmo a inutilização do projeto, possa ser possível o retorno às condições iniciais. Para isso utilizou-se os métodos *Transaction.Start*() para o início do algoritmo de alteração do documento e *Transaction.Commit*() para a finalização da edição.

Segue-se com a captura dos pontos de intersecção de cada elemento construtivo hospedado na parede em questão. A Figura 29 mostra parte do código responsável por localizar (linhas 158 e 159) e calcular os pontos iniciais e finais, superiores e inferiores, dos objetos do tipo “abertura”, da conversão do sistema imperial para o métrico e da localização relativa ao vetor normal à linha central da parede (linhas 166 a 177), e do instanciar a abertura com um componente na lista de estruturas, através do método “add_abertura” (linha 178).

Figura 28 - Código exemplo de cálculo de propriedades

```
157         # tratamento das aberturas
158         inserts = wall.FindInserts(True, True, True, True);
159         for insertId in inserts:
160             e = doc.GetElement(insertId)
161             local = e.Location.Point
162             doorHeight = e.Symbol.get_Parameter(
163                     BuiltInParameter.FAMILY_HEIGHT_PARAM).AsValueString()
164             doorWidth = e.Symbol.get_Parameter(
165                     BuiltInParameter.FAMILY_WIDTH_PARAM).AsValueString()
166             min1 = XYZ(
167                 local.X + (cmtoinch(float(doorWidth))/2)
168                 * estrutura.vectorAlongNormal.X,
169                 local.Y + (cmtoinch(float(doorWidth))/2)
170                 * estrutura.vectorAlongNormal.Y,
171                 local.Z)
172             max2 = XYZ(
173                 local.X - (cmtoinch(float(doorWidth))/2)
174                 * estrutura.vectorAlongNormal.X,
175                 local.Y - (cmtoinch(float(doorWidth))/2)
176                 * estrutura.vectorAlongNormal.Y,
177                 local.Z + (cmtoinch(float(doorHeight))) )
178             estrutura.add_abertura(min1, max2, e.Category.Name)
```

Fonte: Autoria Própria (2022)

A partir destas e de outras informações importantes, passou-se a criação dos elementos do quadro estrutural, estes sendo armazenados em uma estrutura de

objetos chamada “estrutura” através do método “add_componente”. O código da Figura 30 mostra como são definidas as posições da linha de desenho da guia, de seção tipo “U”, superior aos elementos de abertura da parede. A definição da rotação do componente é feita através do parâmetro que informa o tipo da estrutura, como, por exemplo, “TRACK_VERGA” para elementos de verga e “TRACK_CONTRAVERGA” para elementos de verga” (linhas 182 e 186).

Figura 29 - Código exemplo de definição de parâmetros

```
180             # guia superior da abertura
181             li = operator.itemgetter(-1)(estrutura.aberturas)
182             estrutura.add_Component(li.max1, li.max2, TRACK_VERGA)
183
184             # guia inferior da abertura
185             if li.min1.Z > estrutura.min1.Z:
186                 estrutura.add_Component(li.min1, li.min2, TRACK_CONTRAVERGA)
```

Fonte: Autoria Própria (2022)

O mesmo é implementado para o cálculo de posições, quantidades e propriedades dos outros elementos, como os guias inferiores das aberturas, superiores das aberturas do tipo Porta, guias inferiores e superiores do painel construtivo, montantes de perfil “Ce” intermediários, e os montantes iniciais e finais das paredes, os quais devem também considerar as ligações e interações com outras paredes do projeto, suas posições, o ângulo de aplicação em relação ao eixo Z e o distanciamento máximo entre os montantes. Ou seja, foram necessárias as implementações de diversas estruturas de controle e a manipulação de uma grande quantidade de variáveis. A Figura 31 mostra parte deste código responsável pela aplicação dos montantes intermediários. Nela é possível verificar um exemplo de adição de um montante a lista de estruturas que serão posteriormente adicionadas ao documento do projeto arquitetônico (linha 280).

Figura 30 - Exemplo de código de criação de montantes

```
239         #montantes intermediarios
240         anglenew = estrutura.angle
241         naabertura = -9999999
242         topoabertura = 0
243         ehBatente = False
244         aberturaMin1 = 0
245         while math.fabs(
246             currentStartPoint.DistanceTo(endPoint)) > distanceBetweenColumns:
247             nextPoint = XYZ(
248                 currentStartPoint.X + distanceBetweenColumns
249                 * estrutura.vectorAlongNormal.X,
250                 currentStartPoint.Y + distanceBetweenColumns
251                 * estrutura.vectorAlongNormal.Y,
252                 currentStartPoint.Z)
253             min1 = XYZ(nextPoint.X, nextPoint.Y, nextPoint.Z)
254             max2 = XYZ(nextPoint.X, nextPoint.Y, nextPoint.Z
255                 + estrutura.wallWidth)
256 
257             # tratamento das aberturas
258             aberturas = estrutura.aberturas
259             for a in aberturas:
260                 if (a.min1.DistanceTo(estrutura.startPoint)
261                     > a.min2.DistanceTo(estrutura.startPoint)):
262                     temp_a_min1 = a.min2; temp_a_min2 = a.min1;
263                     temp_a_max1 = a.max2; temp_a_max2 = a.max1
264                     ('inverteu ', a.typeVoid)
265                 if (type(aberturaMin1) == int and
266                     currentStartPoint.DistanceTo(
267                         XYZ(temp_a_min1.X,
268                         temp_a_min1.Y,
269                         estrutura.min1.Z)) < (distanceBetweenColumns
270                             + 2*comprimento_montante)) :
271                     aberturaMin1 = temp_a_min1
272                     min1 = XYZ(
273                         temp_a_min1.X,
274                         temp_a_min1.Y,
275                         estrutura.min1.Z)
276                     max2 = XYZ(
277                         temp_a_max1.X,
278                         temp_a_max1.Y,
279                         estrutura.min1.Z + estrutura.wallWidth)
280                     estrutura.add_Component(min1, max2, ON_START_OF_OPEN_COLUMN)
281                     nextPoint = XYZ(
282                         min1.X + distanceBetweenColumns * estrutura.vectorAlongNormal.X,
283                         min1.Y + distanceBetweenColumns * estrutura.vectorAlongNormal.Y,
284                         estrutura.min1.Z)
```

Fonte: Autoria Própria (2022)

Após a definição da lista de estruturas que serão inseridas no projeto, para cada uma destas estruturas foi feita a seleção da família de elemento, a criação do elemento através do método *NewFamilyInstance*() (linhas 317 e 335) e da definição de diversos parâmetros internos da família (como na linha 320), como os níveis de elevação e o ângulo de giro sobre o eixo longitudinal da peça que deve ser inserida com a abertura da seção voltada para direção superior da parede (Figura 31).

Figura 31 - Inserção dos elementos no projeto

```
313  # Cógigo de inserção dos elementos ao projeto e da
314  # criaçao do modelo analítico para ser utilizado futuramente na análise estrutural
315  for comp in estrutura.componentes:
316      if comp.typeStruct.structuralType == Structure.StructuralType.Beam:
317          track = doc.Create.NewFamilyInstance(
318              comp.line, comp.symbol(), estrutura.levelId, comp.typeStruct.structuralType)
319          track.get_Parameter(
320              BuiltInParameter.STRUCTURAL_BEAM_END0_ELEVATION).SetValueString(
321                  str(comp.startPoint.Z))
322          track.get_Parameter(
323              BuiltInParameter.STRUCTURAL_BEAM_END1_ELEVATION).SetValueString(
324                  str(comp.endPoint.Z))
325          track.get_Parameter(
326              BuiltInParameter.Z_JUSTIFICATION).Set(2)
327          track.get_Parameter(
328              BuiltInParameter.Y_JUSTIFICATION).Set(2)
329          track.get_Parameter(
330              BuiltInParameter.STRUCTURAL_BEND_DIR_ANGLE).Set(comp.angle())
331          analitico.add_Component(
332              comp.startPoint, comp.endPoint, ANALYTIC_BEAM)
333      else:
334          if comp.typeStruct.structuralType == Structure.StructuralType.Column:
335              track = doc.Create.NewFamilyInstance(
336                  comp.startPoint, comp.symbol(),
337                  estrutura.levelId, comp.typeStruct.structuralType) #nonestructural?
338              trackType = track.get_Parameter(BuiltInParameter.SLANTED_COLUMN_TYPE_PARAM)
339              trackType.Set(2)
340              track.get_Parameter(
341                  BuiltInParameter.SLANTED_COLUMN_GEOMETRY_TREATMENT_BASE).Set(3)
342              track.get_Parameter(
343                  BuiltInParameter.SLANTED_COLUMN_GEOMETRY_TREATMENT_TOP).Set(3)
344              trackTopOffset = track.get_Parameter(
345                  BuiltInParameter.FAMILY_TOP_LEVEL_OFFSET_PARAM)
346              trackTopOffset.Set(comp.endPoint.Z - estrutura.max1.Z)
347              trackBottomOffset = track.get_Parameter(
348                  BuiltInParameter.FAMILY_BASE_LEVEL_OFFSET_PARAM)
349              trackBottomOffset.Set(comp.startPoint.Z + estrutura.startPoint.Z)
350              trackTopLevel = track.get_Parameter(
351                  BuiltInParameter.FAMILY_TOP_LEVEL_PARAM)
352              wallOffset = wall.get_Parameter(
353                  BuiltInParameter.WALL_TOP_OFFSET)
354              walltoplevel = wall.get_Parameter(
355                  BuiltInParameter.WALL_HEIGHT_TYPE)
356              wallDesconectedHeight = wall.get_Parameter(
357                  BuiltInParameter.WALL_USER_HEIGHT_PARAM)
358              trackTopLevel.Set(walltoplevel.Id)
359              trackBottomLevel = track.get_Parameter(
360                  BuiltInParameter.FAMILY_BASE_LEVEL_PARAM)
361              trackBottomLevel.Set(estrutura.levelId.Id)
362              new_angle = -estrutura.angle + comp.angle()
363              if new_angle >= (2 * math.pi): new_angle -= (2 * math.pi)
364              if new_angle <= (- 2 * math.pi): new_angle += (2 * math.pi)
365              track.get_Parameter(
366                  BuiltInParameter.STRUCTURAL_BEND_DIR_ANGLE).Set(new_angle)
367              analitico.add_Component(comp.startPoint, comp.endPoint, ANALYTIC_COLUMN)
368      track.get_Parameter(BuiltInParameter.ALL_MODEL_INSTANCE_COMMENTS).Set(estrutura.comment)
```

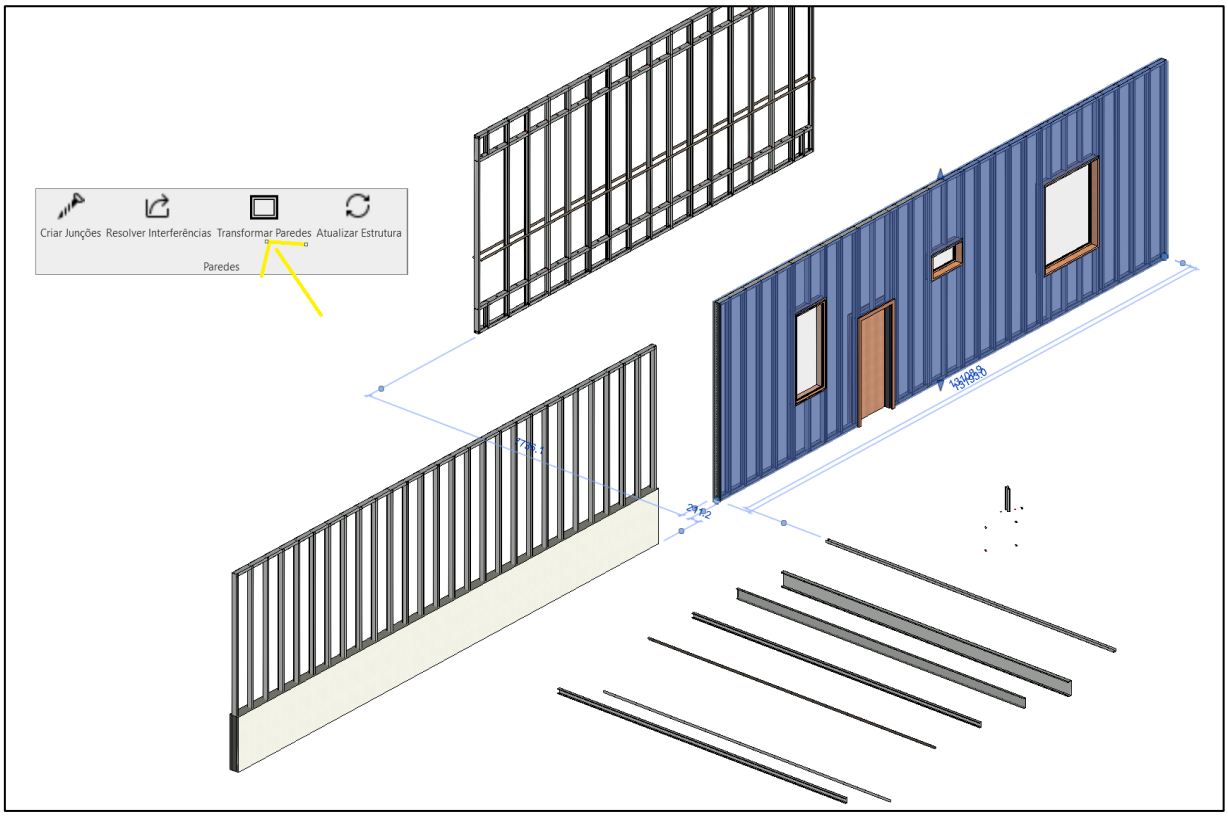
Fonte: Autoria Própria (2022)

Da mesma forma, os códigos de atualização de elementos construtivos, inserção de guias de apoio, de contenção lateral e do tratamento de interferências entre sistemas, foram implementados seguindo-se a mesma metodologia e com resultados semelhantes (Anexo 1). Passou-se então a realização de testes e avaliação dos resultados.

4.2 Testes e Avaliação

Após o desenvolvimento das famílias paramétricas dos componentes e das soluções em forma de *plug-in*, foi possível desenvolver um fluxo de trabalho para modelagem dos painéis estruturais básicos do sistema construtivo, buscando-se viabilizar o projeto com este sistema de forma eficiente, menos trabalhosas que com os modelos atuais que são processos extremamente repetitivos, e sem a necessidade de compra de licenças das poucas, e caras, soluções presentes no mercado.

Figura 32 - Exemplos de Uso



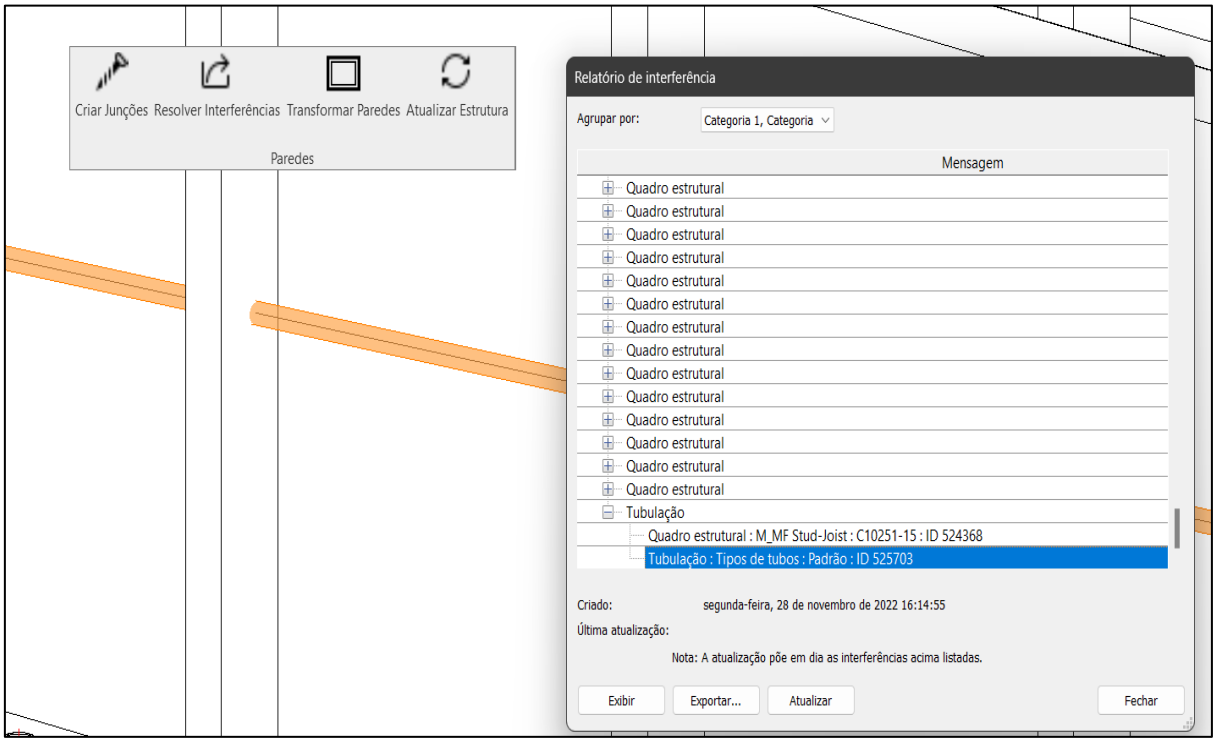
Fonte: Autoria Própria (2022)

Assim, como pode ser verificado na Figura 32, utilizando a ferramenta de

transformar paredes desenvolvida neste trabalho, foi possível modelar diversos tipos de painéis estruturais para vários modelos de paredes arquitetônicas. A figura 32 também apresenta o surgimento de diversas famílias de componentes paramétricos desenvolvidos neste trabalho. Porém as ferramentas de montagem de painéis foram desenvolvidas de forma a possibilitar a utilização de quaisquer famílias de vigas e pilares estruturais do *REVIT* ou modeladas por terceiros seguindo-se os padrões e *templates* de desenvolvimento para estes tipos de famílias.

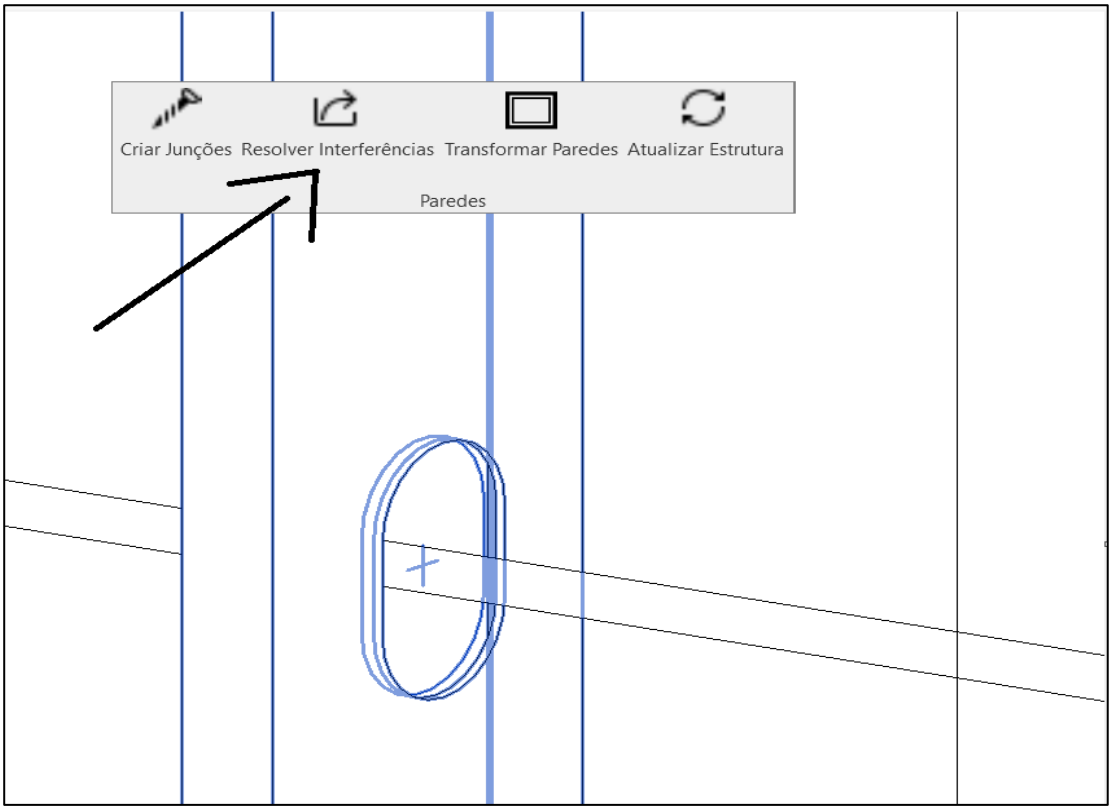
Durante os testes foi possível verificar o funcionamento da ferramenta de resolução de interferências pelo corte da estrutura em locais diversos e de acordo com a forma necessária para resolução da interferência como a apresentada na Figura 33.

Figura 33 - Exemplo de Interferência de projeto



Fonte: Autoria Própria (2022)

Figura 34 - Exemplo de resolução de interferência



Fonte: Autoria Própria (2022)

É possível verificar que uma interferência de tubulação com um montante de pilar estrutural diverso dos desenvolvidos neste trabalho (Figura 33), também pode ser resolvida utilizando-se a ferramenta (Figura 34). É importante ressaltar que esta ferramenta foi modelada de forma a funcionar, além de com as famílias desenvolvidas neste trabalho, com a maioria das famílias de massa sólida disponíveis no *REVIT*.

5 CONCLUSÃO

Assim, foi possível desenvolver e avaliar a utilidade de um conjunto de famílias de *plug-ins* de modelagem de estruturas em *Light Steel Frame* para utilização no *AUTODESK REVIT*, bem como de um conjunto de *plug-in*s de auxílio na montagem dos painéis e resolução de interferências entre os diversos sistemas.

O presente estudo teve seus objetivos atendidos, uma vez que foram apresentados os procedimentos de criação das famílias paramétricas, para o modelo construtivo LSF, sendo criadas as famílias de perfis, guias, montantes e travamentos. Também pela criação de um conjunto de *plug-in*s para o controle da utilização destas famílias na modelagem de painéis estruturais para paredes.

Como forma de avaliação, após a inserção dessas famílias dentro de um modelo de projeto de paredes, foi elaborado um protótipo que permitiu aferir a funcionalidade das famílias perante as questões gráficas, documentais, de obtenção de dados quantitativos de material e na identificação e resolução de interferências entre componentes.

Com as soluções de modelagem desenvolvidas, é possível a elaboração de projetos de fabricação e montagem com materiais e processos mais econômicos, simplificando a estrutura do projeto e possibilitando um maior nível de detalhamento para evitar erros e desperdício de materiais. No entanto, por questões de limitações no tempo de desenvolvimento, o trabalho se limitou à modelagem dos painéis construtivos de um protótipo com uma parametrização simplificada das estruturas das paredes, não abordando uma análise mais aprofundada dos elementos de fixação e outros elementos do sistema construtivo, tais como placas e membranas.

Apesar dessa limitação, os resultados obtidos mostraram que a utilização destas soluções apresentadas para o projeto deste sistema construtivo é útil e permite uma automatização maior no processo de modelagem, evitando também a sobrecarga do projetista e alcançando o objetivo esperado desta pesquisa. Assim, apresentam-se sugestões para trabalhos futuros.

5.1 Sugestões para Trabalhos Futuros

Considerando-se que esse modelo não se apresenta pronto, visto que melhorias a esse e seus elementos podem ser aplicadas, são listadas abaixo como sugestões para trabalhos futuros:

- Elaboração de novos parâmetros para as famílias desenvolvidas, como um controle mais rígido das aberturas e cortes no material;
- Implementação de um modelo analítico, o qual possibilitará a implementação de etapas de cálculo e análise estrutural;
- Modelagem dos painéis de fechamento e dos componentes isolantes;
- Incorporação de tecnologia de inteligência artificial e aprendizado de máquina aos *plug-in*s desenvolvidos;
- Elaboração da documentação completa de montagem, inclusive com a exportação de arquivos CNC para perfiladeiras;
- Modelagem das estruturas de cobertura, forros, pisos e sistemas auxiliares, dentre outros;
- Implementação de uma solução para carregamento automático das famílias necessárias ao projeto e de um instalador para as ferramentas do tipo *plug-in*.

REFERÊNCIAS

AGACAD. **Complete Metal Wall Frame Creator for *REVIT***. [site da web], 2022. Disponível em: <https://agacad.com/products/bim-solutions/metal-framing-wall/overview>. Acesso em: 18 abr. 2022.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 8094**: Material metálico revestido e não revestido - Corrosão por exposição à névoa salina - Método de ensaio. Rio de Janeiro: ABNT, 1983.

_____. **NBR 8800**: Projeto de estruturas de aço e de estruturas mistas de aço e concreto de edifícios. Rio de Janeiro: ABNT, 2008.

_____. **NBR 15758-1**: Sistemas construtivos em chapas de gesso para drywall - Projeto e procedimentos executivos para montagem - Parte 1: Requisitos para sistemas usados como paredes. Rio de Janeiro: ABNT, 2009.

_____. **NBR 15253:** Perfis de aço formados a frio, com revestimento metálico, para painéis estruturais reticulados em edificações — Requisitos gerais. Rio de Janeiro: ABNT, 2014.

_____. **NBR 15498**: Chapas cimentícias reforçadas com fios, fibras, filamentos ou telas - Requisitos e métodos de ensaio. Rio de Janeiro: ABNT, 2021a.

_____. **NBR 14762**: Dimensionamento de estruturas de aço perfis formados a frio. Rio de Janeiro: ABNT, 2021b.

_____. **NBR 14715-1**: Chapas de gesso para drywall - Parte 1: Requisitos. Rio de Janeiro: ABNT, 2021c.

_____. **NBR 16970**: Sistemas construtivos estruturados em perfis leves de aço conformados a frio, com fechamento em chapas delgadas. Rio de Janeiro: ABNT, 2022.

AUTODESK. **Key *REVIT* Concepts**. [site da web], 2022. Disponível em: <https://knowledge.*AUTODESK*.com/support/*REVIT*/learn/caas/qsarticles/key-*REVIT*-concepts.html>. Acesso em: 18 abr. 2022.

BEN, Cezar A. T. D. **Estudo técnico e de mercado do *Light Steel Framing* na cidade de campo mourão**. Monografia (Graduação) do curso de Engenharia Civil. Universidade Tecnológica Federal do Paraná: Campo Mourão, 2016. Disponível em: <https://1library.org/document/ydvxmgjy-estudo-tecnico-mercado-light-framing-cidade-campo-mourao.html>. Acesso em: 22 abr. 2022.

CAMPESTRINI, Tiago Francisco *et al.* **Entendendo BIM**: Uma Visão do Projeto de Construção Sob o Foco da Informação. Curitiba: Tiago Francisco Campestrini, 2015. Disponível em: Acesso em: Abril de 2016.

CARREGARI, Luana. ***Light Steel Frame*** **garante obras rápidas e limpas**. Centro Brasileiro da Construção em Aço. Web, 2019. Disponível em: <https://faststeel.com.br/light-steel-frame-garante-obras-rapidas-e-limpas/>. Acesso em: 22 abr. 2022.

CASTRO, Renata C. M. de. **Arquitetura e tecnologia em sistemas construtivos industrializados**. Dissertação (Mestrado). Engenharia Civil, Universidade Federal de Ouro Preto: Ouro Preto, 2005. Disponível em: <https://www.repositorio.ufop.br/handle/123456789/6246>. Acesso em: 22 abr. 2022.

CONSELHO BRASILEIRO DE CONSTRUÇÃO SUSTENTÁVEL (CBCS). **Diretrizes de Ação CBCS**. Rio de Janeiro, jun. 2013. Disponível em: <http://www.cbcs.org.br/_5dotSystem/userFiles/Sobre CBCS/CBCS_Diretrizes de Acao_rev1.pdf>. Acesso em: 22 abr. 2022.

DOMARASCKI, Conrado S.; FAGIANI, Lucas S. **Estudo comparativo dos Sistemas Steel Frame, Concreto PVC e Sistema Convencional.** Monografia (Graduação) do Curso de Engenharia Civil. Fundação Educacional de Barretos: Barretos, 2009. 76p. Disponível em: <https://www.docsity.com/pt/estudo-comparativo-dos-sistemas-construtivos-steel-fra me-concreto-pvc-e-sistema-convencional/4857952/>. Acesso em: 22 abr. 2022.

DO NASCIMENTO, José Marcos. **A importância da compatibilização de projetos como fator de redução de custos na construção civil.** Revista Online IPOG. Instituto de Pós-Graduação e Graduação IPOG. Goiânia, GO, 2013. Disponível em: <https://d1wqtxts1xzle7.cloudfront.net/38433970/Tecnicas_de_compatibilizacao.-with-cover-page-v2.pdf>. Acesso em 20 abr. 2022.

GATTI, Wagner. **Método Construtivo Steel Frame, Sustentabilidade e Economia na Construção Civil**. Monografia (Bacharelado) do curso de Engenharia Civil. Universidade Alto Vale do Rio do Peixe: Caçador, 2016. Disponível em: <https://acervo.uniarp.edu.br/wp-content/uploads/tccs-graduacao/Metodo-construtivo-Steel-Frame-sustentabilidade-e-economia-na-construcao-civil.-Wagner-Gatti.-2016. pdf>. Acesso em: 22 abr. 2022.

GORGOLEWSKI, M. Developing a Simplified Method of Calculating U-Values in Light Steel Framing. **Building and Environment**, v42, n1, 2006.

GRUBLER, T. H. **Estudo comparativo entre os métodos construtivos *Light Steel Frame*, alvenaria convencional e alvenaria estrutural**. Monografia (Graduação) do curso de Engenharia Civil. UNIJUÍ. Ijuí, 2021. Disponível em: <https://bibliodigital.unijui.edu.br:8443/xmlui/handle/123456789/7220>. Acesso em: 12 jun. 2022.

GUALBERTO, E.. **15 *Softwares* BIM que você precisa conhecer em 2020/2021**. Gerência de Obras [site da web], 2021. Disponível em: <https://gerenciadeobras.com.br/15-*softwares*-bim/>. Acesso em: 18 abr. 2022.

IDHEA – Instituto para o Desenvolvimento da Habitação Ecológica. **Materiais ecológicos e tecnologias sustentáveis para arquitetura e construção civil:** conceito e teoria. Apostila n. 2 Materiais Ecológicos e Tecnologias Sustentáveis. São Paulo, 2006.

JARDIM, G. T. da C; CAMPOS, A. S. **Light Steel Framing:** uma aposta do setor siderúrgico no desenvolvimento tecnológico da construção civil. In: MARTINS, P. *et al.* Inovação em construção civil: monografias. São Paulo, Instituto UNIEMP, 2013. p. 27-45.

KNECHTEL, M. do R.. **Metodologia da pesquisa em educação dialogada:** uma abordagem teórico-prática. Curitiba: Intersaberes, 2014.

LEITE, F. T.. **Métodos e técnicas de pesquisa**. Aparecida: Idéias & Letras, 2008.

MATIAS, L; NUNES, A. F; CRUZ, R. D. **Desperdícios na Construção Civil**. Revista Campo do Saber: Brasil, v. 4, n. 3, 2018. Disponível em: <http://periodicos.iesp.edu.br/index.php/campodosaber/article/view/120>. Acesso em: 28 abr. 2022.

MINISTÉRIO DAS CIDADES (BRASIL). Secretaria Nacional de Habitação. **Diretrizes para Avaliação técnica de produtos:** Diretrizes SINAT nº003, Revisão 1: sistemas construtivos estruturados em perfis leves de aço conformados a frio, com fechamentos em chapas delgadas (Sistemas leves tipo Light Steel Framing). Brasília, 2012.

OLIVEIRA, G. **Análise comparativa entre o sistema construtivo em *Light Steel Framing* e o sistema construtivo tradicional**. Dissertação (Graduação) Eng. Civil UFP: João Pessoa, 2012. Disponível em: repositorio.ufmg.br/bitstream/1843/BUBD-AGWGDM/1/monografia_eduardo_luciano_ufmg_2014_vers_o_final.pdf. Acesso em: 28 abr. 2022.

*REVIT*APIDOCS. ***REVIT* Api Docs.** Site da Web, 2022. Disponível em: <https://www.*REVIT*apidocs.com/>. Acesso em: 27 jun. 2022.

ROVARIS, C. **Estudo para ampliação do uso da madeira para a construção de habitações no Brasil**. 2006. in: RODRIGUES, F. C. Steel Framing: Engenharia. Série Manual da Construção em Aço. Rio de Janeiro: CBCA, 2016. Disponível em: <http://www.skylightestruturas.com.br/downloads/101497_manual_lsf_engenharia_2016.pdf>. Acesso em: 22 abr. 2022.

SANTIAGO, A. K. **O uso do sistema *Light Steel Framing* associado a outros sistemas construtivos como fechamento vertical externo não-estrutural**. Dissertação (Mestrado). Programa de Pós-graduação em Engenharia Civil. Universidade Federal de Ouro Preto: Ouro Preto, 2008. Disponível em: <http://www.repositorio.ufop.br/jspui/handle/123456789/2248>. Acesso em: 26 abr.

2022.

SANTIAGO, A. K.; FREITAS, Arlene M. S.; CASTRO, Renata. C. M. de. **Steel framing**: arquitetura. 2ª ed. Rio de Janeiro: CBCA, 2012. ISBN 978-85-89819-32-9. Disponível em: <https://engprime.com.br/wp-content/uploads/2020/07/Manual_SF_Arquitetura_web. pdf>. Acesso em: 22 abr. 2022.

SENA, P. C. P. de. **Automação de processos de projeto e programação em BIM:** Dynamo, Python e C#. Dissertação (Mestrado em Arquitetura, Urbanismo e Tecnologia) - Instituto de Arquitetura e Urbanismo, USP, São Carlos, 2019. Disponível em: <https://doi.org/10.11606/D.102.2020.tde-12032020-14413>. Acesso em: 28 nov. 2022.

STRUCSOFT. **MWF Pro Metal:** The professional grade light gauge steel framing *software*. [site da web], 2022. Disponível em: <https://strucsoftsolutions.com/mwf-pro-metal/> . Acesso em 18 abr. 2022.

VIVAN, A. L; PALIARI, J. C; NOVAES, C. C. **Vantagem produtiva do sistema light steel framing:** da construção enxuta à racionalização construtiva. XIII Encontro Nacional de Tecnologia do Ambiente Construído, 2010. Disponivel em: <https://www.scribd.com/document/259804362/Vantagem-Produtiva-Do-Sistema-Light-Steel-Framing-Da-Construcao-Enxuta-a-Racionalizacao-Construtiva>. Acesso em: 22 abr. 2022.

YANG, X.; GRUSSENMEYER, P.. Automating Parametric Modelling From Reality-Based Data by *REVIT* Api Development. *In:* Remondino, Fabio; Georgopoulo, A. (org). **Latest Developments in Reality-Based 3D Surveying and Modeling**. Basel: MPDI, 2018.

APÊNDICE A

```
#-*- coding: utf-8 -*-

__title__ = "Transformar Paredes"
__autor__ = "Jean Carlos Triches"
__doc__   = ""

from copy import copy
import os, sys, math, datetime, time, clr, operator
from random import randint
from pyrevit import revit, forms
from Snippets._selection import get_selected_elements
from System.Collections.Generic import List
from Autodesk.Revit.DB import *
from Autodesk.Revit.DB.Structure import *
doc   = __revit__.ActiveUIDocument.Document
uidoc = __revit__.ActiveUIDocument
app   = __revit__.Application

clr.AddReference("System")

PATH_SCRIPT = os.path.dirname(__file__)

distanceBetweenColumns = 1.5

# typeStruct:
class TypeStruct:
    def __init__(
        self, familyName, familyType, structuralType,
        angle = 0,
        category = BuiltInCategory.OST_StructuralFraming):
        self.familyName = familyName
        self.familyType = familyType
        self.category = category
        self.angle = angle
        self.structuralType = structuralType
        elements = (FilteredElementCollector(doc)
            .OfCategory(category)
            .WhereElementIsElementType()
            .ToElements())
        for element in elements:
            elementFamilyName = element.get_Parameter(
                BuiltInParameter.ALL_MODEL_FAMILY_NAME).AsString()
            elementFamilyType = element.get_Parameter(
                BuiltInParameter.ALL_MODEL_TYPE_NAME).AsString()
            if elementFamilyName == (familyName
                and elementFamilyType == familyType):
```

```
                t = Transaction(doc, 'ativando familia')
                t.Start()
                self.symbol = element
                self.symbol.Activate()
                t.Commit()
                break

COLUMN = TypeStruct(
                    'jean_coluna_Ce',
                    'UC305x305x97',
                    Structure.StructuralType.Column,
                    0,
                    BuiltInCategory.OST_StructuralColumns)
END_COLUMN = TypeStruct(
                    'jean_coluna_Ce',
                    'UC305x305x97',
                    Structure.StructuralType.Column,
                    math.pi,
                    BuiltInCategory.OST_StructuralColumns)
ON_START_OF_OPEN_COLUMN = TypeStruct(
                    'jean_coluna_Ce',
                    'UC305x305x97',
                    Structure.StructuralType.Column,
                    math.pi,
                    BuiltInCategory.OST_StructuralColumns)
TOP_TRACK = TypeStruct(
                    'jean_viga4',
                    '362CT125-33',
                    Structure.StructuralType.Beam,math.pi)
BOTTON_TRACK = TypeStruct(
                    'jean_viga4',
                    '362CT125-33',
                    Structure.StructuralType.Beam)
TRACK_VERGA = TypeStruct(
                    'jean_viga4',
                    '362CT125-33',
                    Structure.StructuralType.Beam)
TRACK_CONTRAVERGA = TypeStruct(
                    'jean_viga4',
                    '362CT125-33',
                    Structure.StructuralType.Beam,
                    math.pi)
BEAM = TypeStruct(
                    'jean_viga4',
                    '362CT125-33',
                    Structure.StructuralType.Beam)
STRAP = TypeStruct(
                    'M_MF Strap',
                    '50x1',
```

```
                    Structure.StructuralType.Beam)
HORIZONTAL_BRACE = TypeStruct('jean_viga4',
                    '362CT125-33',
                    Structure.StructuralType.Beam,math.pi)

ANALYTIC_BEAM = TypeStruct(
                    'jean_viga4',
                    '362CT125-33',
                    Structure.StructuralType.Beam,
                    0,
                    BuiltInCategory.OST_AnalyticalMember)
ANALYTIC_COLUMN = TypeStruct(
                    'jean_coluna_Ce',
                    'UC305x305x97',
                    Structure.StructuralType.Column,
                    0,
                    BuiltInCategory.OST_AnalyticalMember)

class StructComponent:
    def __init__(self, startPoint, endPoint, typeStruct):
        self.startPoint = startPoint
        self.endPoint = endPoint
        self.typeStruct = typeStruct
        self.line = Line.CreateBound(startPoint, endPoint)
        self.min1 = XYZ(startPoint.X, startPoint.Y, startPoint.Z)
        self.min2 = XYZ(endPoint.X, endPoint.Y, startPoint.Z)
        self.max1 = XYZ(startPoint.X, startPoint.Y, endPoint.Z)
        self.max2 = XYZ(endPoint.X, endPoint.Y, endPoint.Z)

    def angle(self):
        return self.typeStruct.angle

    def symbol(self):
        return self.typeStruct.symbol

class VoidComponent:
    def __init__(self, startPoint, endPoint, typeVoid):
        self.startPoint = startPoint
        self.endPoint = endPoint
        self.typeVoid = typeVoid
        self.min1 = XYZ(startPoint.X, startPoint.Y, startPoint.Z)
        self.min2 = XYZ(endPoint.X, endPoint.Y, startPoint.Z)
        self.max1 = XYZ(startPoint.X, startPoint.Y, endPoint.Z)
        self.max2 = XYZ(endPoint.X, endPoint.Y, endPoint.Z)

    def angle(self):
        return self.typeStruct.angle

    def symbol(self):
```

```
        return self.typeStruct.symbol

class Struct:
    def __init__(self, startPoint, endPoint,
                    level = 0, wallWidth=1000, comment = ''):
        self.startPoint = self.min1 = startPoint
        self.endPoint = self.min2 = endPoint
        self.vectorAlong = XYZ(endPoint.X - startPoint.X,
            endPoint.Y - startPoint.Y, endPoint.Z - startPoint.Z)
        self.vectorAlongNormal = self.vectorAlong.Normalize()
        self.angle = self.vectorAlongNormal.AngleTo(XYZ.BasisX)
        self.levelId = level
        self.comment = comment
        self.line = Line.CreateBound(startPoint, endPoint)
        self.componentes = []
        self.aberturas = []
        self.wallWidth = wallWidth
        self.max1 = XYZ(startPoint.X, startPoint.Y, wallWidth)
        self.max2 = XYZ(endPoint.X, endPoint.Y, wallWidth)

    def add_Component(self, startPoint, endPoint, typeStruct):
        self.componentes.append(
                StructComponent(startPoint, endPoint, typeStruct))

    def add_abertura(self, startPoint, endPoint, typeVoid):
        self.aberturas.append(
                VoidComponent(startPoint, endPoint, typeVoid))

def get_parameter_value_by_name(element, parameterName):
        return element.LookupParameter(parameterName).AsValueString()

def cmtoinch(number):
    return (number/10)/30.48

H_HORIZONTAL_BRACE = []
H_HORIZONTAL_BRACE.append(cmtoinch(500))
H_HORIZONTAL_BRACE.append(cmtoinch(3500))

if __name__ == "__main__":
    selection = [doc.GetElement(x) for x in
        uidoc.Selection.GetElementIds()]
    for wall in selection:
        startPoint = wall.Location.Curve.GetEndPoint(0)
        endPoint = wall.Location.Curve.GetEndPoint(1)
        level_0 = uidoc.Document.GetElement(wall.LevelId)
        comment = 'p'+str(randint(0,10000))
        comment_analitico = 'A'+str(randint(0,10000))
        wallWidth = cmtoinch(float(get_parameter_value_by_name(wall,
            'Altura desconectada')))
```

```
estrutura = Struct(startPoint , endPoint, level_0, wallWidth,
    comment)
analitico = Struct(startPoint , endPoint, level_0, wallWidth,
    comment_analitico)
print('angulo estrutura ', estrutura.angle)
print ('wallwidth ', wallWidth)

#inicializando a trasação
t = Transaction(doc, 'column')
t.Start()

#marca de comentário na parede
wall.get_Parameter(
    BuiltInParameter.ALL_MODEL_INSTANCE_COMMENTS).Set(comment)

# tratamento das aberturas
inserts = wall.FindInserts(True, True, True, True);
for insertId in inserts:
    e = doc.GetElement(insertId)
    local = e.Location.Point
    doorHeight = e.Symbol.get_Parameter(
            BuiltInParameter.FAMILY_HEIGHT_PARAM).AsValueString()
    doorWidth = e.Symbol.get_Parameter(
            BuiltInParameter.FAMILY_WIDTH_PARAM).AsValueString()
    min1 = XYZ(
        local.X + (cmtoinch(float(doorWidth))/2)
        * estrutura.vectorAlongNormal.X,
        local.Y + (cmtoinch(float(doorWidth))/2)
        * estrutura.vectorAlongNormal.Y,
        local.Z)
    max2 = XYZ(
        local.X - (cmtoinch(float(doorWidth))/2)
        * estrutura.vectorAlongNormal.X,
        local.Y - (cmtoinch(float(doorWidth))/2)
        * estrutura.vectorAlongNormal.Y,
        local.Z + (cmtoinch(float(doorHeight))) )
    estrutura.add_abertura(min1, max2, e.Category.Name)

    # guia superior da abertura
    li = operator.itemgetter(-1)(estrutura.aberturas)
    estrutura.add_Component(li.max1, li.max2, TRACK_VERGA)

    # guia inferior da abertura
    if li.min1.Z > estrutura.min1.Z:
        estrutura.add_Component(
                li.min1, li.min2, TRACK_CONTRAVERGA)

# guia inferior
# depende das portas
```

```
inicio_atual = estrutura.min1
portas = []
for p in estrutura.aberturas:
    if p.typeVoid == 'Portas':
        portas.append(p)
portas.sort(
    key = lambda porta : (
        porta.startPoint.DistanceTo(estrutura.startPoint)))
for p in portas:
    fim = p.min1
    prox_inicio = p.min2
    if (p.min1.DistanceTo(estrutura.startPoint)
        > p.min2.DistanceTo(estrutura.startPoint)):
        fim = p.min2
        prox_inicio = p.min1
    estrutura.add_Component(inicio_atual,
        fim,
        BOTTON_TRACK)
    inicio_atual = prox_inicio
fim = estrutura.min2
estrutura.add_Component(inicio_atual,
    fim,
    BOTTON_TRACK)

# guia superior
estrutura.add_Component(
        estrutura.max1, estrutura.max2, TOP_TRACK)

# travamento horizontal
# depende das aberturas
for h in H_HORIZONTAL_BRACE:
    inicio_atual = XYZ(
                        estrutura.min1.X,
                        estrutura.min1.Y,
                        estrutura.min1.Z + h)
    aberturas = estrutura.aberturas
    aberturas.sort(key = lambda porta :
            porta.startPoint.DistanceTo(estrutura.startPoint))
    for a in aberturas:
        if h > a.min1.Z and h < a.max1.Z:
            fim = XYZ(
                    a.min1.X,
                    a.min1.Y,
                    estrutura.min1.Z + h)
            prox_inicio = XYZ(
                                a.min2.X,
                                a.min2.Y,
                                estrutura.min1.Z + h)
            if (a.min1.DistanceTo(estrutura.startPoint)
```

```
                > a.min2.DistanceTo(estrutura.startPoint)):
                fim = XYZ(
                    a.min2.X,
                    a.min2.Y,
                    estrutura.min1.Z + h)
                prox_inicio = XYZ(
                        a.min1.X,
                        a.min1.Y,
                        estrutura.min1.Z + h)
            estrutura.add_Component(
                    inicio_atual,
                    fim,
                    HORIZONTAL_BRACE)
            inicio_atual = prox_inicio
    fim = XYZ(
            estrutura.min2.X,
            estrutura.min2.Y,
            estrutura.min1.Z + h)
    estrutura.add_Component(
        inicio_atual,
        fim,
        HORIZONTAL_BRACE)

# montantes
comprimento_montante = (
    COLUMN.symbol.LookupParameter("a").AsDouble())
currentStartPoint = estrutura.startPoint
min1 = XYZ(
    currentStartPoint.X,
    currentStartPoint.Y,
    currentStartPoint.Z)
max2 = XYZ(
    currentStartPoint.X,
    currentStartPoint.Y,
    currentStartPoint.Z + estrutura.wallWidth)
CurrentPanelLength = (
    estrutura.endPoint.DistanceTo(estrutura.startPoint))
numberOfColumnsInPanel = (
    math.ceil(CurrentPanelLength / distanceBetweenColumns))

#primeiro montante
estrutura.add_Component(min1, max2, COLUMN)

#montantes intermediarios
anglenew = estrutura.angle
naabertura = -9999999
topoabertura = 0
ehBatente = False
aberturaMin1 = 0
```

```
        while (math.fabs(
            currentStartPoint.DistanceTo(endPoint))
                > distanceBetweenColumns):
            nextPoint = XYZ(
                currentStartPoint.X + distanceBetweenColumns
                * estrutura.vectorAlongNormal.X,
                currentStartPoint.Y + distanceBetweenColumns
                * estrutura.vectorAlongNormal.Y,
                currentStartPoint.Z)
            min1 = XYZ(nextPoint.X, nextPoint.Y, nextPoint.Z)
            max2 = XYZ(nextPoint.X, nextPoint.Y, nextPoint.Z
                + estrutura.wallWidth)

            # tratamento das aberturas
            aberturas = estrutura.aberturas
            for a in aberturas:
                if (a.min1.DistanceTo(estrutura.startPoint)
                    > a.min2.DistanceTo(estrutura.startPoint)):
                    temp_a_min1 = a.min2; temp_a_min2 = a.min1;
                    temp_a_max1 = a.max2; temp_a_max2 = a.max1
                    ('inverteu ', a.typeVoid)
                if (type(aberturaMin1) == int and
                    currentStartPoint.DistanceTo(
                        XYZ(temp_a_min1.X,
                        temp_a_min1.Y,
                        estrutura.min1.Z)) < (distanceBetweenColumns
                            + 2*comprimento_montante)) :
                    aberturaMin1 = temp_a_min1
                    min1 = XYZ(
                        temp_a_min1.X,
                        temp_a_min1.Y,
                        estrutura.min1.Z)
                    max2 = XYZ(
                        temp_a_max1.X,
                        temp_a_max1.Y,
                        estrutura.min1.Z + estrutura.wallWidth)
                    estrutura.add_Component(
                            min1,
                            max2,
                            ON_START_OF_OPEN_COLUMN)
                    nextPoint = XYZ(
                        (min1.X + distanceBetweenColumns
                            * estrutura.vectorAlongNormal.X),
                        (min1.Y + distanceBetweenColumns
                            * estrutura.vectorAlongNormal.Y),
                        estrutura.min1.Z)

                    while (nextPoint.DistanceTo(estrutura.min1)
                        < XYZ(
```

```
                    temp_a_min2.X,
                    temp_a_min2.Y,
                    estrutura.min1.Z)
                    .DistanceTo(estrutura.min1)
                    - 2*comprimento_montante):
                min1 = XYZ(
                    nextPoint.X,
                    nextPoint.Y,
                    temp_a_max1.Z)
                max2 = XYZ(
                    nextPoint.X,
                    nextPoint.Y,
                    estrutura.min1.Z
                    + estrutura.wallWidth)
                estrutura.add_Component(
                    min1,
                    max2,
                    COLUMN)
                if temp_a_min1.Z > estrutura.min1.Z:
                    min1 = XYZ(
                        nextPoint.X,
                        nextPoint.Y,
                        estrutura.min1.Z)
                    max2 = XYZ(
                        nextPoint.X,
                        nextPoint.Y,
                        temp_a_min1.Z)
                    estrutura.add_Component(
                        min1,
                        max2,
                        COLUMN)
                nextPoint = XYZ(
                    (nextPoint.X + distanceBetweenColumns
                        * estrutura.vectorAlongNormal.X),
                    (nextPoint.Y + distanceBetweenColumns
                    * estrutura.vectorAlongNormal.Y),
                    estrutura.min1.Z)

            min1 = XYZ(
                temp_a_min2.X,
                temp_a_min2.Y,
                estrutura.min1.Z)
            max2 =  XYZ(
                temp_a_max2.X,
                temp_a_max2.Y,
                estrutura.min1.Z + estrutura.wallWidth)
            aberturaMin1 = 0

        if type(aberturaMin1) == int:
```

```
            estrutura.add_Component(min1, max2, COLUMN)

        currentStartPoint = XYZ(
            min1.X,
            min1.Y,
            estrutura.min1.Z)

    #montante final
    currentStartPoint = estrutura.endPoint
    min1 = XYZ(
        currentStartPoint.X,
        currentStartPoint.Y,
        currentStartPoint.Z)
    max2 = XYZ(
        currentStartPoint.X,
        currentStartPoint.Y,
        currentStartPoint.Z
            + estrutura.wallWidth)
    estrutura.add_Component(
        min1,
        max2,
        END_COLUMN)

    # Cógigo de inserção dos elementos ao projeto e da
    # criaçao do modelo analítico para ser utilizado
    # futuramente na análise estrutural
    for comp in estrutura.componentes:
        if (comp.typeStruct.structuralType
                == Structure.StructuralType.Beam):
            track = doc.Create.NewFamilyInstance(
                comp.line, comp.symbol(),
                estrutura.levelId, comp.typeStruct.structuralType)
            track.get_Parameter(
                (BuiltInParameter
                .STRUCTURAL_BEAM_END0_ELEVATION)).SetValueString(
                        str(comp.startPoint.Z))
            track.get_Parameter(
                (BuiltInParameter
                .STRUCTURAL_BEAM_END1_ELEVATION)).SetValueString(
                    str(comp.endPoint.Z))
            track.get_Parameter(
                BuiltInParameter.Z_JUSTIFICATION).Set(2)
            track.get_Parameter(
                BuiltInParameter.Y_JUSTIFICATION).Set(2)
            track.get_Parameter(
                (BuiltInParameter
                .STRUCTURAL_BEND_DIR_ANGLE)).Set(comp.angle())
            analitico.add_Component(
                comp.startPoint, comp.endPoint, ANALYTIC_BEAM)
```

```
else:
    if (comp.typeStruct.structuralType
        == Structure.StructuralType.Column):
        track = doc.Create.NewFamilyInstance(
            comp.startPoint, comp.symbol(),
            estrutura.levelId,
            comp.typeStruct.structuralType)
        trackType = track.get_Parameter(
            BuiltInParameter.SLANTED_COLUMN_TYPE_PARAM)
        trackType.Set(2)
        track.get_Parameter(
            (BuiltInParameter
            .SLANTED_COLUMN_GEOMETRY_TREATMENT_BASE)).Set(3)
        track.get_Parameter(
            (BuiltInParameter
            .SLANTED_COLUMN_GEOMETRY_TREATMENT_TOP)).Set(3)
        trackTopOffset = track.get_Parameter(
            BuiltInParameter.FAMILY_TOP_LEVEL_OFFSET_PARAM)
        trackTopOffset.Set(
            comp.endPoint.Z - estrutura.max1.Z)
        trackBottomOffset = track.get_Parameter(
            BuiltInParameter.FAMILY_BASE_LEVEL_OFFSET_PARAM)
        trackBottomOffset.Set(
            comp.startPoint.Z + estrutura.startPoint.Z)
        trackTopLevel = track.get_Parameter(
            BuiltInParameter.FAMILY_TOP_LEVEL_PARAM)
        wallOffset = wall.get_Parameter(
            BuiltInParameter.WALL_TOP_OFFSET)
        walltoplevel = wall.get_Parameter(
            BuiltInParameter.WALL_HEIGHT_TYPE)
        wallDesconectedHeight = wall.get_Parameter(
            BuiltInParameter.WALL_USER_HEIGHT_PARAM)
        trackTopLevel.Set(walltoplevel.Id)
        trackBottomLevel = track.get_Parameter(
            BuiltInParameter.FAMILY_BASE_LEVEL_PARAM)
        trackBottomLevel.Set(estrutura.levelId.Id)
        new_angle = -estrutura.angle + comp.angle()
        if new_angle >= (2 * math.pi):
            new_angle -= (2 * math.pi)
        if new_angle <= (- 2 * math.pi):
            new_angle += (2 * math.pi)
        track.get_Parameter(
            (BuiltInParameter
            .STRUCTURAL_BEND_DIR_ANGLE)).Set(new_angle)
        analitico.add_Component(
            comp.startPoint,
            comp.endPoint,
            ANALYTIC_COLUMN)
track.get_Parameter(
```

```
            (BuiltInParameter
            .ALL_MODEL_INSTANCE_COMMENTS)).Set(estrutura.comment)

    #Código de exportação para o FTOOL
    ftool = []
    ftool.append('400 0')
    ftool.append('206')
    ftool.append('1')
    ftool.append('0 1 2 9 2 9 7 0 0 0 0')
    ftool.append('+1.00000e+030 -1.00000e+030')
    ftool.append('-1.00000e+030 +1.00000e+030 '
        +'-1.00000e+030 +1.00000e+030')
    ftool.append('1 1')
    ftool.append('1 0 1 0 1 1 1 0 0 0 0 0 0 0 0')
    ftool.append('0 0')
    ftool.append('3 131075 4 33 65 129 147969 1025 2049 257')
    ftool.append('131075 131081 131089 4097 8193 65545 131075')
    ftool.append('4 7 7 2 8 2 8 4 0 2')
    ftool.append('12 0 0 9 9 7 7 4 2 6')
    ftool.append('1 1')
    ftool.append('0 1 1 1 1 0')
    ftool.append('1 1 0')
    ftool.append('1')
    ftool.append("'Load Case 01' 1")
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0 0 0 0 0')
    ftool.append('0 0 0')
    ftool.append('0')
    ftool.append('0 0 0 0')
    ftool.append(' 0')

    for comp in analitico.componentes:
        ftool.append('2 1 1 1 5 4 4 8 5 1 7')
        ftool.append("{:.5e}".format(comp.max2.X/10)
            + " {:.5e}".format(comp.max2.Y/10))
        ftool.append('0')
        ftool.append("{:.5e}".format(comp.min1.X/10)
            + " {:.5e}".format(comp.max2.X/10)
            + " {:.5e}".format(comp.min1.Y/10)
            + " {:.5e}".format(comp.max2.Y/10))
```

```
    ftool.append('0 0 0 0 0 1')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0')
    ftool.append('0 0 0 0 0')
    ftool.append('0 0 0')
    ftool.append('0')
    ftool.append('0 0 0 0')
    ftool.append(' 0')

#os.remove("analitico.ftl")
with open("analitico.ftl", "w") as f:
    for l in ftool:
        f.write(l+'\n')

t.Commit()
```