MARINA PIRES IASBIK

ANÁLISE DA INTEGRAÇÃO DA MODELAGEM ALGORÍTMICO-PARAMÉTRICA COM A MODELAGEM DA INFORMAÇÃO DA CONSTRUÇÃO (BIM)

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Arquitetura e Urbanismo, para obtenção do título de *Magister Scientiae*.

Orientadora: Andressa C. Pena Martinez

VIÇOSA - MINAS GERAIS
2022

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T

Iasbik, Marina Pires, 1993-
I11i Análise da integração da modelagem algorítmico-paramétrica com a modela-
2022 gem da informação da construção (BIM) / Marina Pires Iasbik. - Viçosa, MG, 2022.
1 dissertação eletrônica (118 f.): il. (algumas color.).

Orientador: Andressa Carmo Pena Martinez.
Dissertação (mestrado) - Universidade Federal de Viçosa, Departamento de Arquitetura e Urbanismo, 2022.
Inclui bibliografia.
DOI: https://doi.org/10.47328/ufvbbt.2022.381
Modo de acesso: World Wide Web.

1. Modelagem de informação da construção. 2. Construção civil - Simulação (Computadores). 3. Arquitetura - Projetos e plantas. I. Martinez, Andressa Carmo Pena, 1982-. II. Universidade Federal de Viçosa. Departamento de Arquitetura e Urbanismo. Programa de Pós-Graduação em Arquitetura e Urbanismo. III. Título.

CDD 22. ed. 690.0285

Bibliotecário(a) responsável: Bruna Silva CRB-6/2552

MARINA PIRES IASBIK

ANÁLISE DA INTEGRAÇÃO DA MODELAGEM ALGORÍTMICO-PARAMÉTRICA COM A MODELAGEM DA INFORMAÇÃO DA CONSTRUÇÃO (BIM)

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Arquitetura e Urbanismo, para obtenção do título de *Magister Scientiae*.

APROVADA: 25 de fevereiro de 2022.

Assentimento:

Marina Pires Iasbik
Marina Pires Iasbik
Autora

Andressa C. P. Martinez
Andressa Carmo Pena Martinez
Orientadora

AGRADECIMENTOS

A Deus.

Aos meus pais, José Álvaro e Selma, por serem o maior exemplo de amor e de bondade na minha vida, por me incentivarem, acreditarem no meu potencial e estarem sempre ao meu lado.

Aos meus irmãos, Felipe, Mateus e Lucas, por nunca terem medido esforços pra me apoiar e ajudar em todos os momentos que precisei, por me mostrarem que ser irmão vai muito além dos nossos laços sanguíneos. Vocês, o pai e a mãe são as maiores riquezas da minha vida.

A toda minha família, por serem alegria e aconchego, é um verdadeiro privilégio ser rodeada do amor de vocês.

Ao Amadeu, meu companheiro de todos os dias, o abraço que me tranquiliza, a paz que sempre sonhei, obrigada por sempre acreditar em mim, mais até do que eu mesma.

Aos meus amigos de infância, as pessoas que eu escolhi para amar há 15, 20 anos, e que continuam ao meu lado. Em cada oportunidade de encontro eu transbordo de amor e felicidade. Obrigada Gabi, por se fazer presente todos os dias, mesmo à distância, sempre querendo meu bem, me apoiando e incentivando em tudo. À Debrinha, minha irmãzinha e confidente da vida toda, que nenhuma distância jamais vai separar.

A todos os outros amigos, aos da faculdade, do mestrado, do intercâmbio, todos vocês fizeram e fazem meus dias mais felizes. Obrigada à Lau pela amizade verdadeira durante a graduação, que se estende para sempre, sem você eu sei que tudo teria sido muito mais difícil.

À minha orientadora, Andressa Martinez, por me inspirar com tanto conhecimento e força de vontade para ir atrás dos objetivos, pelas palavras que também foram incentivo e conforto nos momentos difíceis.

À Universidade Federal de Viçosa (UFV), ao Departamento de Arquitetura e Urbanismo (DAU) e ao Programa de Pós-Graduação em Arquitetura e Urbanismo (PPG.au) pelo apoio e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela concessão da bolsa de estudos.

Ao longo da história da arquitetura, a representação essencial dos edifícios tem sido feita através de desenhos. A leitura de muitos livros mostra como o desenho e o croqui são parte integrante do pensamento e do trabalho criativo do arquiteto. [...] A substituição dos desenhos por uma nova base de representação para o projeto, comunicação e construção dos edifícios, é uma mudança revolucionária e que marca época, tanto na Arquitetura como na Indústria de Construção em geral. Estas mudanças alteram as ferramentas, os modos de construção e os processos de trabalho.

(EASTMAN, 2006, p.2)

RESUMO

IASBIK, Marina Pires, M.Sc., Universidade Federal de Viçosa, fevereiro de 2022. **Análise da integração da modelagem algorítmico-paramétrica com a Modelagem da Informação da Construção (BIM).** Orientadora: Andressa Carmo Pena Martinez.

Esta pesquisa investiga a adaptabilidade do processo de projeto em BIM (*Building Information Modeling*) às novas perspectivas do *design* computacional, com ênfase na integração das lógicas de Modelagem da Informação da Construção (BIM) com a modelagem algorítmico-paramétrica, aqui representada pela modelagem desenvolvida através do *Grasshopper*. Sabe-se que, somadas, essas estratégias de projeto representam a possibilidade de uma grande liberdade e variabilidade na descrição das representações formais, em um modelo coordenado de todos os aspectos de um edifício. No entanto, constatou-se uma incapacidade de importar ou integrar facilmente em BIM elementos geométricos ou conjuntos gerados no *Grasshopper*. Visando buscar alternativas que facilitassem na interoperabilidade BIMxGrasshopper, foram realizados testes a partir de diferentes *software,* plataformas e *plugins* atuais relacionados à transmissão de dados em BIM. Foram escolhidos dois *plugins* visando a conexão do *Grasshopper* ao *Archicad* e ao *Revit*: o *Archicad Live Connection* e o *Rhino Inside Revit*, respectivamente. Comparando-se os fluxos de trabalho relacionados a cada combinação de ferramentas, levantaram-se os pontos positivos e negativos de cada um e foram delimitadas as situações em que cada fluxo é mais adequado. Os testes demonstraram que ainda existem barreiras na integração do BIM com aplicativos de modelagem algorítmico-paramétrica, sobretudo devido às diferenças em suas lógicas de funcionamento, por serem ferramentas de empresas proprietárias distintas. No entanto, algumas soluções foram encontradas para facilitar o processo de projeto, alcançando-se maior interoperabilidade entre software diferentes, e não deixando que as limitações impostas por algumas barreiras, que naturalmente surgiram com os novos processos de projeto a partir do uso do BIM, limitassem o uso de software específicos para cada necessidade de projeto.

Palavras-chave: BIM (*Building Information Modeling*). Modelagem paramétrica. Modelagem algorítmica.

ABSTRACT

IASBIK, Marina Pires, M.Sc., Universidade Federal de Viçosa, February, 2022. **Integration of Building Information Modeling (BIM) logics with algorithmic-parametric modeling.** Adviser: Andressa Carmo Pena Martinez.

This research investigates the adaptability of design process in BIM (Building Information Modeling) to new perspectives of computational design, with emphasis on the integration of Building Information Modeling (BIM) logics with algorithmic-parametric modeling, represented here by a model developed on Grasshopper. It is known that, together, these design strategies represent the possibility of great freedom and variability in the description of formal representations, in a coordinated model of all aspects of a building. However, there was an inability to easily import or integrate geometric elements or sets generated in Grasshopper into BIM. In order to seek alternatives that facilitate BIMxGrasshopper interoperability, tests were carried out using different software, platforms and plugins related to data transmission in BIM. Two plugins were chosen to connect Grasshopper to Archicad and Revit: Archicad Live Connection and Rhino Inside Revit, respectively. Comparing the workflows related to each combination of tools, the positive and negative points of each were identified and the situations in which each flow is more suitable were delimited. The tests showed that there are still barriers in the integration of BIM with algorithmic-parametric modeling applications, mainly due to differences in their operating logic, as they are tools from different proprietary companies. However, some solutions were found to facilitate the design process, achieving greater interoperability between different software, and not letting the limitations imposed by some barriers that naturally emerged with the new design processes from the use of BIM limit the use of specific software for each project need.

Keywords: BIM (Building Information Modeling). Parametric design. Algorithmic design.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

3DM – Formato de modelo 3D nativo do *Rhinoceros*

3DS – Formato de modelo 3D nativo do 3D *Estudio Max*

AEC – Arquitetura, Engenharia e Construção

ALC - *Archicad Live Connection*

API - *Application Programming Interface*

BDS - *Building Description System*

BIM - *Building Information Modeling*

CAD - Computer Aided Design

CG-BIM - Comitê Gestor do BIM

CSV - *Comma-Eeparated Values*

DAG - Directed acyclic graph

DWG – *AutoCAD* DraWinG format

DXF - *Drawing Exchange Format*

GDL - Geometric Description Language

GG – Geometry Gym

GH – *Grasshopper*

GLIDE - para *Graphical Language for Interactive Design*

IFC - *Industry Foundation Classes*

JSON - JavaScript Object Notation

MDIC - Ministério da Indústria, Comércio Exterior e Serviços

MOU - Memorandum Of Understanding

MS – Mantis Shrimp

NURBS - *Non Uniform Rational Basis Spline*

PLN - Formato de modelo 3D nativo do *Archicad*

RIR - *Rhino Inside Revit*

RVT – Formato de modelo 3D nativo do *Revit*

SIGraDI - *Sociedad Iberoamericana de Grafica Digital*

SKP - Formato de modelo 3D nativo do *SketchUp*

STEP – Formato de modelo 3D (Padrão para o Intercâmbio de Dados de Produtos)

TPL – *Textual Programing Language*

VPL - *Visual Programing Language*

XML - Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento ou utilização de ferramentas computacionais para a resolução de problemas na arquitetura é uma prática conhecida na área acadêmica desde a década de 1960 (MITCHELL *et al.,* 1987). A partir da rápida evolução dessas ferramentas, Oxman (2014) delineia uma perspectiva de revolução na teoria da arquitetura, na qual o "*design*", podendo ser traduzido como o processo de projeto, torna-se um importante tema de pesquisa. A partir dos anos 2000, o *design* passa a ser visto também como um meio de produção de conhecimento, promovendo uma mudança teórica seguindo uma abordagem mais científica e baseada na computação.

A introdução de novas tecnologias sugere novos modos de conceber a arquitetura. Desta forma, entende-se que as fases iniciais da concepção projetual são afetadas por esses adventos tecnológicos, que no caso desta pesquisa, estão relacionadas aos aspectos formais.

Em suas investigações sobre métodos de *design*, Oxman (2006) aplica um sistema de modelos de processo de projeto digital e diferencia o conceito de *Digital Design*, baseado em imitar através do computador o processo de projeto baseado no papel, de um conceito no qual arquiteto passa a atuar também como criador de ferramentas de personalização da mídia de projeto, devido à crescente sofisticação da mídia de projeto e sua capacidade de funcionar de maneira integrada e interativa. A esse, a autora posteriormente denominou *Design* Computacional (OXMAN, 2014; OXMAN, 2017; CAETANO e LEITÃO, 2020).

O *design* computacional engloba diferentes tipos de tecnologias desenvolvidas no intuito de auxiliar na prática arquitetônica. Como destacam Caetano e Leitão (2020), a comunidade arquitetônica tem conduzido pesquisas em diversas áreas de *design*, visando diminuir quaisquer limitações sentidas pelos profissionais. Com o tempo, as tecnologias estudadas são aprimoradas e se tornam comercialmente viáveis, podendo ser adotadas pela prática arquitetônica.

A programação para resolução de problemas na arquitetura é uma prática conhecida na área acadêmica desde a década de 1960 (Burry, 1997). Nessa área, o desenvolvimento das linguagens de programação visual (VPL) vem sendo cada vez mais explorado por arquitetos não-programadores, devido à maior facilidade de aprendizagem, quando comparada às linguagens de programação textual (LANDIM, 2019).

Nesse contexto, surge o *Grasshopper*, um *plugin* baseado em VPL e modelagem paramétrica, adicionado ao *Rhinoceros*. Este, tipicamente abreviado por *Rhino*, é um *software* comercial de computação gráfica 3D e de desenho assistido por computador (CAD) desenvolvido por Robert McNeel & Associates. O *Grasshopper* intensificou o interesse pelas técnicas de *design* paramétrico e algorítmico entre a comunidade arquitetônica, principalmente por fornecer uma solução para a automatização de tarefas sem a necessidade de escrever código textual, sendo, portanto, mais intuitiva para usuários não programadores, além de economicamente acessível (LEITÃO E CAETANO, 2020).

O *design* paramétrico pode ser resumido como uma abordagem que descreve a representação de um projeto, simbolicamente baseada no uso de parâmetros (CAETANO; SANTOS; LEITÃO, 2020). O *design* algorítmico é definido por Oxman (2017) como a codificação "de instruções explícitas" para gerar "formas digitais". Como exemplo de *design* algorítmico e paramétrico, pode-se citar um algoritmo desenvolvido através de um código visual no *Grasshopper*, que gera uma fachada com base em um conjunto de parâmetros, como suas dimensões gerais e o tamanho e distribuição dos diferentes elementos da fachada, como aberturas, volumes e revestimentos.

Conforme Oxman (2017) há uma necessidade ampliada de exploração da singularidade de métodos, técnicas, mídias e ferramentas do projeto no campo dos estudos de *design*, a fim de entender o impacto do *design* paramétrico e do *design* algorítmico no surgimento de novas formas de pensar sobre o processo de projeto na arquitetura.

Isso pode ser exemplificado no decorrer do trabalho do arquiteto: a exploração de diferentes ferramentas que auxiliem na modelagem computacional durante as etapas iniciais do processo de projeto, pode auxiliar profissionais a desenvolverem, com certa facilidade, geometrias simples ou complexas. No entanto, os *software* usados para criar essas geometrias muitas vezes não são os mesmos usados para o desenvolvimento e a entrega do projeto em estágio posterior. Para essas fases de execução mais avançadas, os projetos podem ser melhor gerenciados por meio de *software* de Modelagem da Informação da Construção (BIM) (MILLER, STASIUK, 2017).

O BIM (*Building Information Modeling*) pode ser apresentado como um dos principais representantes da evolução do *design* computacional. Succar (2009) o conceitua como um conjunto integrado de políticas, processos e tecnologias, que gera uma

metodologia para gerenciar o projeto e os seus dados em formato digital, auxiliando no processo de projeto, construção e operação, ao longo do ciclo de vida do edifício. A metodologia BIM está presente, por exemplo, em dois *software* amplamente utilizados na prática de projeto contemporânea, o *Graphisoft Archicad* e o *Autodesk Revit*.

De acordo com Caetano e Leitão (2020), as tecnologias relacionadas ao BIM foram desenvolvidas para superar as limitações dos desenhos técnicos 2D em CAD, incorporando dimensões adicionais para melhorar a coordenação do projeto através de um modelo parametricamente conectado. Essas dimensões se traduzem nas informações adicionadas a um banco de dados vinculado ao modelo 3D, que possibilitam, de forma mais ágil, além da visualização da volumetria, avaliações detalhadas de custos, simulações e análises de desempenho, planejamento da construção e maior integração das equipes de projeto. As modificações do projeto são processadas automaticamente em planilhas de custos, nas representações de plantas baixas, dos demais planos e documentos da construção, possibilitando um incremento significativo na qualidade da comunicação e, consequentemente, na qualidade do produto final, a edificação.

Eastman *et al.* (2014) consideram que o BIM se fundamenta em duas tecnologias: o design paramétrico e a interoperabilidade. Nesse contexto, o design paramétrico é associado ao controle das relações de diversos parâmetros editáveis atribuídos aos elementos de um modelo, conforme Veloso, Vasconselos e Scheeren:

> O *design* paramétrico, em linhas gerais, pode ser definido como um processo de *design* que requer a atribuição de definições explícitas que permitem um nível de controle do projeto em desenvolvimento por meio da indicação de relações entre as partes que podem ser editadas a partir de um conjunto de parâmetros - elemento variável e fator quantificável capaz de configurar um sistema de relações (VELOSO, VASCONSELOS E SCHEEREN, 2017, p. 91).

Em um comparativo com o método convencional de projeto, Woodbury (2010) destaca que a definição de relações do *design* paramétrico exige um esforço inicial, para se estabelecer a lógica do conjunto e se explicitar as ideias através de notações formais, mas com o benefício posterior de garantir alterações sem a necessidade de refazer as partes.

A interoperabilidade, de acordo com Eastman *et al.* (2014), é a capacidade dos aplicativos em trocar dados entre si, facilitando os fluxos de trabalho e, algumas vezes, a automatização durante os processos de projeto. Leitão e Caetano (2020)

complementam sobre a importância da interoperabilidade em um contexto de troca de dados relacionados aos diferentes *software* de setores e empresas atuantes no desenvolvimento, construção e operações de um empreendimento, e às diversas etapas envolvidas no processo de projeto.

Comparativamente aos sistemas paramétricos BIM, os algoritmos desenvolvidos no *Grasshopper* possibilitam maior ênfase no desenvolvimento de múltiplas possibilidades formais, que atendam aos parâmetros estabelecidos pelo projetista. Essa exploração formal geralmente ocorre nas etapas iniciais do processo de projeto. No entanto, é de grande interesse que ela também possa ser modificada ao longo de todas as etapas do processo projetual, de forma que seja possível realizar alterações no modelo a qualquer momento e que essas alterações sejam automaticamente transmitidas a todos os planos e tabelas previamente desenvolvidos.

Conforme a complexidade e a abundância de informações interdisciplinares aumentam e são inseridas em modelos BIM, se faz necessário desenvolver soluções que tratem tais informações de forma adequada e as considerem, efetivamente, no processo de projeto. Assim, surgem novas competências para que diferentes profissões relacionadas a AEC atuem de maneira eficiente. Dentre tais competências, Holzer (2015) destaca a programação computacional como algo que ainda possui um papel que está sendo descoberto na arquitetura. Trata-se de um campo interdisciplinar entre a arquitetura e a tecnologia da informação, destacada por alguns escritórios renomados na área de arquitetura, como Arup, Foster & Partners e Zaha Hadid, que têm investido em contratar programadores, visando desenvolver soluções específicas para concepção projetual (CELANI *et al*., 2015).

Tem-se, portanto, duas estratégias de *design*: o paramétrico e algorítmico, que se complementam - Para atender a necessidades formais específicas, a estratégia de parametrização algorítmica, implementada através de recursos da linguagem de programação visual, pode ser eficaz para ampliar os recursos de modelagem formal de *software* baseado em BIM. Por outro lado, *software* baseados em VPL não possuem o mesmo desempenho que os *software* baseados em BIM para a organização e gestão do projeto, adição de informações, controle de todas as etapas do processo, das equipes envolvidas, da conceção até mesmo à demolição de um edifício. Somadas, essas estratégias representam a possibilidade de uma grande liberdade e variabilidade na descrição das representações formais, em um modelo coordenado de todos os aspectos de um edifício.

O problema é que a maioria das soluções BIM não são adequadas para lidar de maneira nativa com projetos algorítmicos, traduzindo-se em uma incapacidade de importar ou integrar facilmente elementos geométricos ou conjuntos gerados em outro *software*. Em outras palavras, a dificuldade de compatibilidade entre *software* BIM e *software* algorítmico (aqui representado pelo *Grasshopper*) é ainda um problema na prática arquitetônica contemporânea, fazendo-se fundamental explorar soluções que possibilitem a melhoria dessa integração.

Conforme Miller (2016), embora o *design* computacional exista graças à tecnologia, ele é considerado uma metodologia de solução de problemas. O arquiteto centrado em projetar a partir do pensamento computacional deve ser capaz de descrever problemas de projeto e os fluxos de trabalho para auxiliar em suas resoluções. A partir das informações abordadas, considerou-se como um problema relacionado às práticas de *design* contemporâneas a integração entre *software* BIM e *software* de modelagem algorítmico-paramétrica baseado em linguagem de programação visual.

Para investigar essas situações-problema e propor soluções, esse trabalho analisa alguns fluxos de trabalho durante o processo de projeto, baseados na integração dos mecanismos e ferramentas de geração de formas de modeladores paramétricos BIM. Utilizam-se sistemas de modelagem algorítmico-paramétrica do *Grasshopper* integrados ao *Archicad* e ao *Revit* para obter melhores resultados na geração, exploração, manipulação e edição de formas livres.

Para a investigação de um fluxo de trabalho baseado na compatibilidade entre *software* BIM (*Archicad* e *Revit*) e de modelagem algorítmica VLP (*Grasshopper*), os principais *plugins* testados foram o *Archicad Live Connection e o Rhino Inside Revit*. Esses *plugins* funcionam como *add-on*[1] para o *Archicad* e para o *Revit,* respectivamente: carregam o *Rhinoceros* e seus outros *plugins* (por exemplo, o *Grasshopper*) na memória do computador, fazendo-os funcionar como qualquer outro *add-on* dentro do *Archicad* ou do *Revit*. Na interface do *Grasshopper*, é adicionada uma coleção de novos componentes para interagir com o *software* BIM em questão, além de permitir

[1] Extensões, complementos ou *plugins* - programa de computador usado para adicionar funções a outros programas maiores. Um *software* pode utilizar *add-ons* visando permitir que desenvolvedores de *software* externos estendam as funcionalidades do produto, suportem funcionalidades antes desconhecidas, reduzam o tamanho do programa ou, até mesmo, separem o código fonte de diferentes componentes devido a incompatibilidade de licenças de *software*. Fonte: https://www.rhino3d.com/inside/revit/1.0/

acesso às duas APIs de *software* (*Application Programming Interfaces*) usando seus componentes de *script*.

Acredita-se que integração de alternativas computacionais pode estimular a diversidade de recursos criativos que subsidiam os projetistas na reflexão, crítica e escolha das melhores soluções durante o processo de projeto. A partir disso, faz-se relevante a exploração de diferentes *software, plataformas e plugins* atuais.

1.1. Objetivos

O objetivo desta pesquisa é investigar a adaptabilidade do processo de projeto em BIM frente às novas perspectivas do *design* computacional, com ênfase nos problemas que envolvem a integração das lógicas de modelagem algorítmico-paramétrica, aqui representada pela modelagem desenvolvida no *Grasshopper*, com a Modelagem da Informação da Construção (BIM).

Destaca-se como objetivo específico:

- Desenvolver testes através de fluxos de trabalho *Grasshopper* x BIM, com foco na modelagem de geometrias durante a concepção formal do projeto em BIM, e utilizando-se de diferentes *software*, *plugins* e plataformas atuais disponíveis (sobretudo o *Archicad Live Connection* e o *Rhino Inside Revit),* para solucionar problemas de transmissão de dados do *Grasshopper* ao *Revit* ou ao *Archicad*.

1.2. Metodologia

Vaishnavi e Kuechler (2005) consideram a DSR (*Design Science Research)* um método capaz de auxiliar na compreensão dos fenômenos artificiais, criados pelo homem. Segundo Hevner *et al.* (2004), numa abordagem pragmática, a DSR não anseia alcançar verdades últimas, grandes teorias ou leis gerais, mas procura identificar e compreender os problemas do mundo real e propor soluções apropriadas, úteis, fazendo avançar o conhecimento teórico da área.

De acordo com Lukka (2003), a DSR se baseia nas seguintes etapas de investigação: (1) Encontrar um problema na prática; (2) Examinar o potencial de investigação junto ao setor alvo; (3) Obter conhecimento geral e profundo do tema; (4) Criar uma solução inovadora e desenvolver um artefato; (5) Implementar a solução e testar; (6) Refletir sobre a aplicabilidade; (7) Identificar e analisar as contribuições teóricas.

Nesse contexto, de acordo com o autor, a definição de artefato é ampla, podendo-se exemplificar diversos tipos de artefatos: constructos, modelos, métodos, diagramas, planos, estruturas organizacionais, entre outros. March e Smith (1995) conceituam o método como um tipo de artefato baseado em um conjunto de passos ou guia para desempenhar uma tarefa específica. Essa tarefa pode estar ligada a teorias ainda incipientes, porém que sejam representações úteis e capazes de explorar um curso para resolução de um problema.

As investigações aqui desenvolvidas se relacionam à interoperabilidade durante o desenvolvimento formal de um projeto, a partir da utilização de diferentes ferramentas necessárias para a execução dos papéis de cada um. Nesse cenário, tais papéis são delegados com o auxílio de *plugins* de modelagem algorítmico-paramétrica de conexão *Grasshopper* x *Archicad/Revit*.

Utilizou-se parte da DSR para auxiliar no desenvolvimento da metodologia proposta, detalhada através de seis etapas de investigação:

(1) Encontrar um problema na prática

Objetivando fazer a caracterização do problema, sua classificação e sua definição, foi realizada uma revisão bibliográfica inicial a respeito da evolução do *Design* Computacional. Nesse contexto, entende-se o BIM e sua relação com o processo de projeto, utilizando-se, de forma prática, diferentes *software*, plataformas e *plugins* para complementar as potencialidades de modelagem da metodologia BIM.

Ferreira e Leitão (2015) relatam que as interfaces básicas de *software* não contemplam os graus de flexibilidade e abstração necessários para atender as demandas específicas dos projetistas, com dificuldades relacionadas à geometria complexa, tarefas repetitivas e exploração de diferentes soluções. Além disto, os *software* funcionam como uma caixa preta, dotada de funcionalidades restritas, o que impede que o usuário interaja com a informação em situações de maior complexidade, devido às limitações inerentes e necessidade de *inputs* muito específicos (AISH, 2013b).

Constatou-se ainda que o desenvolvimento de edifícios com formas mais complexas, através do uso de plataformas baseadas em programação, como o *Grasshopper*, tem influenciado cada vez mais na necessidade de maior agilidade dos arquitetos no manuseio de geometrias durante a concepção formal do projeto e de alterações das formas a qualquer momento, o que é facilitado quando essas alterações estão relacionadas aos diversos parâmetros pré-estabelecidos pelo arquiteto. Nesse

cenário, normalmente os *software* usados para criar geometrias não são os mesmos usados para o desenvolvimento e a entrega do projeto em estágio posterior.

Conforme Silva *et al*. (2019), como os *software* comerciais não atendem muitas das demandas específicas dos projetistas, os mesmos ficam dependentes da utilização de funcionalidades pouco flexíveis, o que limita processos de tomada de decisão e exploração do *design*. Esta situação é agravada quando grandes plataformas desenvolvem funcionalidades exclusivas, as quais não funcionam em conjunto com *software* de outros desenvolvedores, o que impede a interoperabilidade e, portanto, pode agravar as limitações existentes, uma vez que impossibilita o emprego de algumas das premissas do BIM nos processos.

No contexto abordado nesse trabalho, considera-se a relevância da utilização do *Grasshopper* e do *Archicad*/*Revit* durante o processo de projeto. Mas a maioria das soluções BIM não são adequadas para lidar de maneira nativa com projetos algorítmico-paramétricos, traduzindo-se em uma incapacidade de importar ou integrar facilmente em BIM elementos geométricos ou conjuntos gerados em outra plataforma (no caso, o *Grasshopper*).

Portanto, a inserção BIM nas práticas de *design* contemporâneas, onde também se tem o uso de VPLs em *software* de modelagem algorítmico-paramétrica, aponta para os problemas de interoperabilidade na transmissão de informações desses *software* (aqui representados pelo *Grasshopper*) para os *software* BIM (aqui representados pelo *Archicad* e *Revit*).

(2) Obter conhecimento geral e profundo sobre o tema

Essa fase é refletida na fundamentação teórica, realizada através da revisão de livros, artigos científicos, sites, teses e periódicos, com a finalidade de acumular o conhecimento sobre o estado da arte do problema. Tratou-se dos temas correlatos ao trabalho: *design* computacional, com foco no *design* paramétrico e no *design* algorítmico; BIM e suas relações com o design paramétrico; Linguagens de programação visual e as relações entre modelagens algorítmico-paramétricas e a Modelagem de Informação da Construção (BIM). Por fim, tem-se um dos pilares do BIM, utilizado no delineamento dos problemas apontados: necessidade de interoperabilidade nos fluxos de informações de um modelo geométrico entre diferentes *software*.

Essa etapa também contou com a capacitação nos principais *software* relacionados à pesquisa, visando maior conhecimento prático para o entendimento de suas

potencialidades e limitações: *Revit*, *Archicad*, *Rhinoceros*, *Grasshopper*, bem como em diversos plugins de interoperabilidade através desses *software*, com destaque para o *Archicad Live Connection* e para o *Rhino Inside Revit*.

(3) Examinar o potencial de investigação

No Brasil, ainda não está muito disseminado o uso de ferramentas computacionais como apoio à etapa criativa do projeto, embora estas tenham sua potencialidade evidenciada por diversos autores nacionais e internacionais, tanto em Projeto Auxiliado por Computador (CAD) (ARAUJO e CELANI, 2016), como em BIM (JANSSEN *et al.*, 2016).

Além disso, conforme Love *et al.* (2015), a maioria das pesquisas e publicações relativas ao BIM ainda se concentra nas etapas avançadas do projeto: gestão, execução e operação. Cabe destacar, no entanto, que em processos baseados em BIM, não existe uma ordem fixa para as etapas de projeto, ou seja, é possível e desejável que alterações formais também possam ser desenvolvidas de maneira simplificada em qualquer etapa avançada do processo de projeto.

Portanto, há um crescente interesse na exploração das etapas de modelagem formal durante o processo de projeto, a chamada etapa criativa.

Constatou-se também que duas importantes estratégias de *design* abordadas na evolução do *design* computacional – o *design* paramétrico e algorítmico – podem se complementar durante o processo de projeto: tem-se, por um lado, as necessidades de modelagem formal, que podem ser auxiliadas pela parametrização algorítmica, desenvolvida, por exemplo, no Grasshopper. No entanto, o *Grasshopper* é apenas uma ferramenta para a execução dos códigos visuais propostos, a fim de se obter maior liberdade formal. Já os *software* baseados em BIM possibilitam a organização e gestão do projeto, adição de informações ao modelo, controle de todas as etapas do processo, das equipes envolvidas, da concepção até mesmo à demolição de um edifício. Somando-se os recursos algorítmicos aos recursos propiciados por *software* baseado em BIM, tem-se a possibilidade de uma grande liberdade e variabilidade na descrição das representações formais, em um modelo coordenado de todos os aspectos de um edifício.

De acordo com Silva *et al*. (2019), visando à independência do projetista do futuro perante os processos BIM, é necessário atribuir ao arquiteto a função de criador de suas soluções. Isto deve ser feito respeitando o progresso e aprendizado acerca

de programação, uma vez que estas habilidades representam um paradigma que é encarado como uma barreira de abstração (AISH, 2013b).

Investigações acerca dos problemas relacionados à integração das lógicas de modelagem algorítmico-paramétrica, aqui representada pela modelagem desenvolvida no *Grasshopper*, com a Modelagem da Informação da Construção (BIM) podem, portanto, auxiliar no desenvolvimento de estratégias de troca de informações entre diferentes *software*, tornando o processo de projeto mais eficiente.

(4) Investigar e testar alternativas para os problemas constatados

Não houve intenção de criar um artefato inovador ou método específico para a resolução dos problemas que serão exemplificados. No entanto, essa pesquisa utilizou uma implementação prática para triangular as análises realizadas com a revisão da literatura.

A revisão bibliográfica levantou alternativas para solução dos problemas mencionados, presentes em ferramentas da era do Design Computacional que operam utilizando VPL. Essas ferramentas podem funcionar como extensões das grandes plataformas BIM, o que amplia sua capacidade em manipular os parâmetros existentes, aumentando as possibilidades de criação e desenvolvimento de geometrias complexas e o gerenciamento de informações do modelo.

Partindo dessas alternativas, como forma de investigação de seu funcionamento, foram analisados alguns fluxos de trabalho *Grasshopper* x *Archicad*/*Revit*, exemplificando o uso das plataformas e os recursos disponíveis. O intuito foi de investigar a adaptabilidade do processo de projeto em BIM frente aos problemas que envolvem a integração de processos de *design* paramétrico e algorítmico através de diferentes *software* de projeto. Esses estudos tiveram em comum a conexão de modelos desenvolvidos no *Grashopper* com o *Archicad* e com o *Revit*, a partir do uso dos *plugins Archicad Live Connection* e *Rhino Inside Revit*, respectivamente.

As alternativas investigadas foram testadas em um modelo já desenvolvido em *Grasshopper*, que seria modificado para a adequação às demandas dos *plugins* testados. Esse estudo foi transformado em um artigo, apresentado no capítulo 5, e publicado no SIGraDI 2020 (Congresso anual da *Sociedad Iberoamericana de Gráfica Digital*). A metodologia para o artigo está melhor detalhada no capítulo 5, mas pode ser resumida a partir de três etapas:

(a) Foi realizado um levantamento de *software* e *plugins* para a transmissão de dados entre plataformas de projeto algorítmico-paramétrico e *software* BIM. Como opção de fluxo de trabalho para a transmissão de dados, adotou-se o modelo de importação através de *plugins*.
(b) A conversão foi totalmente executada no *Grasshopper* para os dois *software* BIM mais utilizados atualmente: para o *Archicad,* através do *plugin Archicad Live Connection* (ALC), e para o *Revit,* através do *plugin Rhino Inside Revit* (RIR). Esses *plugins* foram escolhidos por serem os únicos *plugins* testados com conexão em tempo real com os *software* BIM, uma vez que operam bidirecionalmente, mantendo as propriedades intrínsecas do modelo *BIM*, sem a necessidade de parar a exploração formal, exportar a geometria e importar para o novo *software*.
(c) O modelo algorítmico-paramétrico utilizado para os testes foi desenvolvido inteiramente no *Grasshopper*. Apesar de alguns *software* BIM possuírem nativamente possibilidades de modelagem algorítmico-paramétrica, como o caso do par *Autodesk Revit-Dynamo*, o ambiente *Rhinoceros-Grasshopper* foi adotado por ser um dos mais disseminados na atualidade, além de ter sido considerado mais eficiente no desenvolvimento formal e na conexão com diversos outros *plugins*, elevando a inteligência de modelagem da arquitetura para além do manuseio da geometria.

(5) Refletir sobre a aplicabilidade

Nesta etapa, verifica-se se as ferramentas utilizadas em todos os testes desenvolvidos possuem capacidade de integração com *software* BIM, ou seja, se a transmissão de informações entre os diferentes *software*, plataformas e *plugins* selecionados ocorre de forma eficiente. Analisam-se as lógicas de programação de cada *software* e as relações possíveis entre eles, comparando-se *software* BIM baseado em modelagem associativa (*Archicad*/*Revit*) e *software* CAD baseado em fluxo de dados (*Grasshopper*). Verifica-se se o código visual original necessitou de alterações, retirada ou adições de componentes para a geração correta de geometria nos *software* BIM.

(6) Identificar e analisar as contribuições teóricas

Elencam-se pelo menos cinco questões a serem esclarecidas:

- Houve contribuições para o processo de projeto? Quais foram?

- Os fluxos de trabalho propostos de fato auxiliam para que profissionais de diferentes setores possam utilizar diferentes *software*, todos integrados a um mesmo modelo central em BIM?
- Os fluxos de informação utilizados na pesquisa contribuem para o desenvolvimento formal e para os processo de colaboração e gestão de projeto em BIM?
- É possível verificar melhorias na qualidade projetual?
- Os fluxos têm potencialidade para gerar novas pesquisas correlatas à área de estudo?

1.3. Estrutura do trabalho

Essa dissertação está organizada em seis capítulos:

- Neste capítulo 1, **Introdução**, apresenta-se um panorama geral de conceituação do campo de estudo e a problemática abordada. Delineiam-se os principais objetivos da pesquisa e a metodologia *Design Science Research* (DSR) utilizada, no intuito de solucionar os problemas encontrados e cumprir com os objetivos levantados;
- No segundo capítulo, **Fundamentação Teórica**, discute-se o referencial teórico dos principais eixos temáticos desta pesquisa: (2.1) Evolução do *Design* Computacional; (2.2) *Design* paramétrico; (2.3) *Design* algorítmico, (2.4) Linguagens de Programação Visual (VPL); (2.5) BIM - *Building Information Modeling*; e conceitos relacionados ao BIM e às problemáticas anteriormente mencionadas: (2.6) Interoperabilidade e (2.7) Colaboração e gestão de projetos em BIM;
- O terceiro capítulo, ***Software*, Plataformas e *Plugins* utilizados**, faz um levantamento das principais ferramentas utilizadas para os testes desenvolvidos ao longo da pesquisa: *Archicad*; *Revit*; *Rhinoceros* e seu *plugin* algorítmico *Grasshopper*; e os principais *plugins* para conexão *Grasshopper* x BIM;
- O quarto capítulo, **Fluxos de trabalho BIM algorítmico-paramétricos,** inicia a aplicação prática das teorias estudadas. Analisam-se os fluxos de trabalho integrados entre *Rhinoceros*, *Grasshopper*, *Archicad* e *Revit*, e os *plugins Archicad Live Connection e Rhino Inside Revit.*
- O quinto capítulo, ***Integration of BIM and algorithmic design logics through data exchange between Grasshopper plugin and Revit and Archicad software***, consiste em um artigo publicado no SIGraDI 2020 (Congresso anual da

Sociedad Iberoamericana de Gráfica Digital). Nele é feito um levantamento de estratégias atuais de conversão entre *software* de modelagem algorítmico-paramétrica e BIM (*Grasshopper x Archicad/Revit*), além da aplicação prática das teorias analisadas, com a migração de um *script* em *Grasshopper* para o *Archicad* e para o *Revit*, comparando-se os fluxos de trabalho nos dois *software* BIM e levantando-se diretrizes de projeto a partir dos testes desenvolvidos.

- Por fim, o sexto capítulo, **Considerações Finais,** gera uma discussão a partir do trabalho desenvolvido.

2. FUNDAMENTAÇÃO TEÓRICA

Esta pesquisa está inserida no contexto do *design* computacional, com enfoque na metodologia de Modelagem de Informação da Construção (BIM). Dentro desse contexto, considera-se fundamental compreender melhor os conceitos de *design* paramétrico e *design* algorítmico e analisar situações de integração das capacidades paramétricas de *softwar*e BIM com plataformas algorítmico-paramétricas baseadas em linguagem de programação visual (VPL), como o *Grasshopper*.

Como enfatizam Montesinos, Galant e Pascual (2018), a complementação dos potencias BIM com os recursos de ferramentas como o *Grasshopper* começou a ocorrer na base do processo de produção da construção devido à necessidade, em BIM, de novas possibilidades de exploração do projeto. Um dos focos dessa pesquisa é a potencialidade de explorações formais que o uso em conjunto dessas ferramentas proporciona. Soma-se a isso a possibilidade de associação das geometrias geradas no *Grasshopper* a uma variedade de *plugins* operáveis nessa plataforma, o que possibilita análises de dados climáticos, análises de modelos energéticos, simulações estruturais, entre outros potenciais. Esses recursos, quando associados à inteligência BIM, amplificam as possibilidades de soluções de projetos.

O desenvolvimento de edifícios com formas mais complexas, através do uso de plataformas baseadas em programação, tem influenciado cada vez mais na necessidade de maior agilidade dos arquitetos no manuseio de geometrias durante a concepção formal do projeto e de alterações das geometrias a qualquer momento, o que é facilitado quando essas estão relacionadas aos diversos parâmetros pré-estabelecidos pelo arquiteto.

Na realidade, a arquitetura baseada em BIM não divide as etapas do projeto da mesma forma que a arquitetura tradicional. Em BIM, tem-se um modelo central, e diversos setores envolvidos de forma colaborativa no projeto o atualizam constantemente a partir de seus modelos locais, com novas informações relacionadas à divisão de funções pré-estabelecida.

Com isso, as alterações realizadas a qualquer momento são transmitidas automaticamente a toda documentação gerada, e o projeto evolui de acordo com os diferentes níveis de necessidades de representações dessas documentações. Isso se traduz em economia de tempo para os ajustes e, consequentemente, de custos. Interoperar o *Grasshopper* ao *Archicad* ou ao *Revit*, portanto, representa a automação ainda

mais eficiente dos processos relacionados às geometrias dos edifícios, e a aplicação dos resultados ao modelo central, ao qual todos os setores envolvidos no projeto podem ter acesso e editar.

No entanto, como abordam Miller e Stasiuk (2017), durante o processo de transmissão de dados de uma plataforma de design algorítmico-paramétrico, como o *Grasshopper*, para BIM (*Archicad*/*Revit*), em projetos de maior complexidade, existe uma inadequação do *software* de modelagem algorítmico-paramétrica no suporte às estruturas de dados BIM. Esse atrito pode gerar desperdícios durante o processo de projeto, visto que pode resultar documentação incompleta. Com isso, é necessário, muitas vezes, remodelar as geometrias em BIM, devido à baixa fidelidade dos modelos com relação aos modelos anteriormente desenvolvidos no *Grasshopper*.

A necessidade de a equipe de projeto “mascarar” ou “desenhar à mão” vários objetos por desenho é um processo de consumo de tempo que contradiz diretamente utilidades fundamentais do BIM para a entrega do projeto, como economia de tempo, organização dos dados, maior eficiência na manipulação durante a etapa projetual e em todo o ciclo de vida da edificação.

Apesar disso, Larrondo (2017) reafirma a importância da busca pela integração entre *software* de modelagem VPL e *software* BIM. O primeiro fornece um ambiente de programação visual para definir a lógica algorítmica e gerar o modelo, que, a partir de *plugins* adequados, poderá ser utilizado e editado também em um *software* BIM. Essa integração permite que programas baseados em código tradicional (em linguagem de programação *Python*, *Design Script*, etc.) sejam escritos através de códigos visuais. Muitas das formas complexas que são difíceis de gerar no *Archicad* ou no *Revit,* portanto, podem ser feitas usando um *software* VPL, como é o caso do *Grasshopper*.

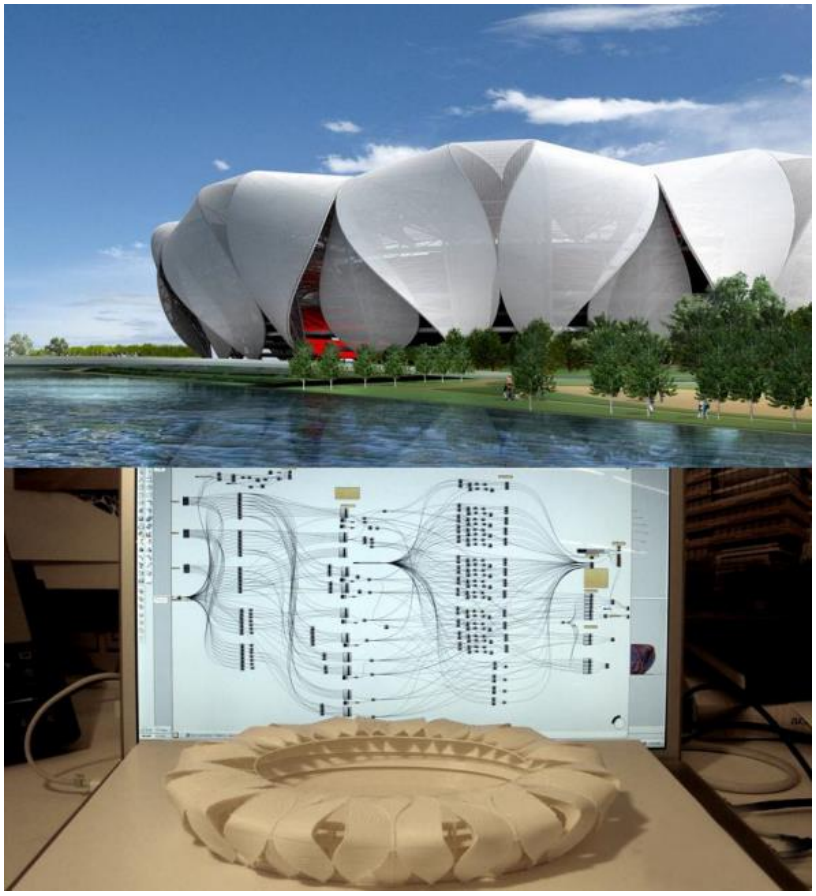
Nesse capítulo será revisada a evolução histórica dos métodos de *design* computacional, visando investigar sobre o processo de projeto em BIM e suas relações com o *design* algorítmico e com o *design* paramétrico, aqui representado por um processo de modelagem em um *software* de programação visual, o *Grasshopper*. Além disso, como mencionado, no processo de projeto em BIM, a interoperabilidade entre diferentes *software* e plataformas é fundamental, sendo também relevante um maior aprofundamento deste conceito.

2.1. Evolução do *Design* Computacional

O impacto da computação em todas as áreas do conhecimento humano tem redefinido os alicerces da sociedade contemporânea. Manuel Castells (1996) chamou este fenômeno de “Sociedade da Informação”. Todos os setores da sociedade estão se modificando em um ritmo acelerado, imersos em um período histórico caracterizado por uma revolução centrada nas tecnologias de informação.

A prática arquitetônica não ficou de fora dessas transformações. Observa-se uma demanda cada vez maior, em edifícios contemporâneos, por geometrias complexas e altamente personalizáveis, com ampla variedade de formas e dimensões. O resultado dessas demandas se reflete no uso crescente de uma variedade de ferramentas computacionais, desde o final do século XX. O *Hangzhou Stadium* (Figura 1), por exemplo, foi desenvolvido pela NBBJ com o auxílio do *software Rhinoceros*, *plugin Grasshoper* e prototipagem digital.

Figura 1 - Concepção formal do *Hangzhou Stadium*, da NBBJ



Fonte: Davidson (2015).

Essas ferramentas são usadas não só nos sistemas de produção e construção, mas também nos domínios de criação e de desenvolvimento do projeto arquitetônico. A administração mais consciente de ferramentas que auxiliem a prática de projeto possibilita maior precisão e, consequentemente, qualidade na representação e execução de um modelo arquitetônico (BRAVO MARTÍNEZ, 2015).

Em meados do século XX, em um contexto pós Segunda Guerra e em resposta ao grande impulso da industrialização e da produção em massa, alguns pesquisadores começaram a buscar um maior entendimento sobre “métodos de *design*” na arquitetura, ou seja, “métodos de projeto”. Movimentos como o *Design Methods*, em 1960 e o *Design Thinking*, em 1969, geraram importantes contribuições relacionadas ao programa arquitetônico, à inteligência artificial e à aplicação de técnicas computacionais para solucionar os problemas de projeto e compor as formas dos objetos (BRIGITTE, 2019).

Os fundamentos conceituais de sistemas BIM se deram concomitantemente aos primeiros estudos sobre o *design* computacional. Em 1962, Douglas C. Engelbart demonstra sua visão do futuro da arquitetura em seu artigo *Augmenting Human Intellect*, que já sugeria o *design* baseado em objetos, manipulação paramétrica e um banco de dados relacional, características primordiais da metodologia BIM tal qual se conhece atualmente:

> O arquiteto começa a inserir uma série de especificações e dados - um piso de laje de seis polegadas, paredes de concreto de doze polegadas, com 2,5 metros de altura dentro da escavação e assim por diante. Quando ele termina, a cena revisada aparece na tela. Uma estrutura está tomando forma. Ele o examina, ajusta. (...) Essas listas crescem em uma estrutura interligada cada vez mais detalhada, que representa o pensamento amadurecido por trás do *design* real (ENGELBART, 1962, p.5).

Bruce Archer (1965, p.51), em seu "Método Sistemático para Designers" desenvolvia a relação do *design thinking* com a gestão de projeto: "Aproxima-se rapidamente o tempo em que as técnicas de tomada de decisões de *design* e de gestão terão tanto em comum que uma se tornará nada mais que a extensão da outra".

Em *Notes on the Synthesis of Form*, Christopher Alexander (1964), influenciava uma escola inicial de cientistas da computação de programação orientada a objetos. Ele propôs que o processo de projeto fosse menos arbitrário e mais lógico, vinculando

a forma ao seu contexto, favorecendo a racionalidade na tomada de decisão (ALEXANDER, 1964). Mais tarde, seu trabalho serviu como base conceitual para o desenvolvimento de importantes subsídios das arquiteturas computacionais, como a parametrização, as gramáticas da forma e os sistemas generativos (NATIVIDADE, 2010).

Alexander (1964) também contribuiu estabelecendo a condição do programa de necessidades na definição de variáveis, bem como desenvolvendo um processo de projeto sistemático baseado na teoria de probabilidades e grafos, cuja aplicação dependia do auxílio computacional (BRIGITTE, 2019). Por mais pensativos e robustos que esses sistemas fossem, as estruturas conceituais não poderiam ser realizadas sem uma interface gráfica por meio da qual interagir com esse modelo de construção.

O início da interação homem-computador se deu na mesma década, com os primeiros passos no desenvolvimento das interfaces gráficas. Mitchell (1977) relata que, em 1963, Ivan E. Sutherland desenvolveu uma “máquina de desenhar”, em sua tese de doutorado (MIT), denominada *Sketchpad*. Essa ferramenta possibilitou o desenho interativo de elementos geométricos sobre um monitor de computador, sendo então considerado o primeiro programa CAD, que inspiraria o desenvolvimento de uma interface muito mais sofisticada, por William M. Newman, em 1966. Ahlquist e Menges (2011) atribuem a Sutherland importantes contribuições sobre métodos de variação de *design*, restrições de *design* e instâncias paramétricas.

Somente na década de 1970 a prática da arquitetura auxiliada por computador começaria efetivamente (CAETANO, SANTOS E LEITÃO, 2020), fundamentada por estudos de importantes nomes do período, como Eastman (1975), March e Steadman (1971) e Mitchell (1977). Kalay (2004) considera essa como a Primeira Geração dos sistemas CAD de projeto.

No entanto, devido à robustez exigida pelos equipamentos e ao alto custo agregado, o uso dos computadores estava restrito aos centros de pesquisa, como o dirigido por Charles M. Eastman no *Institute for Physical Planning at Carnegie Mellon University*. Nesse centro, Eastman desenvolveu um dos primeiros projetos a implementar com sucesso um modelador com base de dados específica, o *Building Description System* (BDS). De acordo com Eastman *et al.* (2014), esse foi considerado o primeiro *software* a descrever elementos individuais de biblioteca que podem ser recuperados e adicionados a um modelo. Se utiliza de uma interface gráfica do usuário, visualizações ortográficas e em perspectiva e um banco de dados classificável que

permite ao usuário recuperar informações categoricamente por atributos, incluindo tipo de material e fornecedor.

Posteriormente, esse sistema foi modificado para *Graphical Language for Interactive Design* (GLIDE), que se baseou em uma linguagem de programação capaz de suportar operações geométricas em descrições paramétricas de edifícios (BRIGITTE, 2019).

No campo conceitual, Mitchell (1977) destaca que a grande evolução deste período foi a alteração da visão sobre o papel do computador no processo de projeto. Inicialmente visto como um substituto, ele passa para o status de “assistente inteligente”, cuja expectativa envolvia a execução de tarefas mais triviais, liberando o arquiteto para tarefas mais importantes, como a tomada de decisões.

Nos anos 80, o *design* computacional tornou-se um campo reconhecido e aceito na arquitetura, a partir do avanço das construções e, sobretudo, com a comercialização das primeiras ferramentas CAD (Caetano, Santos e Leitão, 2020). Kalay (2004) considera essa a segunda geração dos sistemas CAD de projetos, que se distingue pelo advento dos computadores pessoais, com a redução dos custos das tecnologias agregadas. Além disso, enquanto os sistemas da primeira geração foram introduzidos voltados para o projeto arquitetônico propriamente dito, na segunda geração a ênfase estava no desenho e na modelagem (KALAY, 2004).

A terceira geração, a partir da década de 90 (KALAY, 2004), é marcada por retomar o apoio aos processos de projeto arquitetônico, como na primeira geração, e não apenas desenvolver a representação de edifícios. De acordo com Caetano, Santos e Leitão (2020), nesse período, os avanços obtidos em duas décadas de pesquisa ultrapassaram a automação de tarefas de desenho e proporcionaram sistemas CAD mais inteligentes, consolidando e desenvolvendo processos de projeto digital.

Ainda nos anos 90, o *design* computacional se mostrava um campo estabelecido, com suas próprias conferências e periódicos. Também nessa década a popularidade do *software* CAD aumentou entre os arquitetos, principalmente devido à automação de tarefas repetitivas, o que levava ao aumento da produtividade.

Na quarta geração, dos anos 2000 até os dias atuais, a mudança de paradigma tornou-se mais acentuada. Conforme Oxman (2006), nesse período a exploração da forma passa a ocorrer principalmente através do meio digital, implicando em mudanças significativas na metodologia do processo arquitetônico. Nesse contexto observa-se a integração de diferentes técnicas baseadas em computação, como simulação de

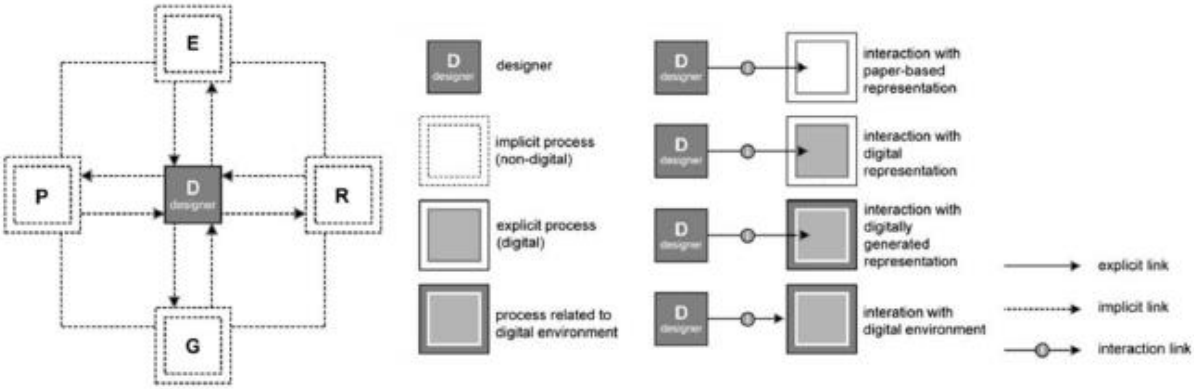
construção, otimização evolutiva, programação orientada a objeto, inteligência artificial, sistemas de gerenciamento de banco de dados, novos métodos de fabricação e a parametrização, originando, portanto, novos termos e abordagens de *design* (OXMAN, 2017).

Com relação às novas abordagens, Janssen, Chen e Mohanty (2016) explicam sobre a transição de modelagens convencionais, como em *Autocad* ou *Rhinoceros* para modelagens BIM, como *Archicad* ou *Revit*. Nesse processo, a forma de pensar o projeto se modifica, pois existe um ponto chave, quando o arquiteto precisa mudar de ferramentas. Antes deste ponto, entidades planas como paredes e lajes de piso podem ser modeladas como polígonos simples, e elementos lineares como colunas e vigas podem ser definidos como polilinhas (em sistemas CAD). Uma vez que a mudança é feita para as ferramentas BIM, as entidades precisam ser definidas com mais precisão como elementos de construção, exigindo que sejam espessadas e que os materiais e outros detalhes sejam definidos.

De um lado tem-se os sistemas BIM, fundamentados pelos conceitos do *design* paramétrico. Por outro lado, em sistemas CAD, como no *Rhinoceros*, com o *Grasshopper* atuando em conjunto, se potencializam as aplicações do *design* algorítmico. Kolarevic (2003) considera que, nesta nova geração, o arquiteto deixa de modelar formas para articular uma lógica interna de geração da forma. A partir de uma maior exploração e experimentação do *design*, o arquiteto pode produzir, de maneira automática, uma gama de possibilidades formais, dentre as quais ele pode escolher a mais apropriada para ser desenvolvida.

Oxman (2006) aponta a necessidade de distinção entre o conceito de Projeto Assistido por Computador, basicamente baseado em imitar o processo baseado no papel, e de Projeto Arquitetônico Digital - *Digital Architectural Design*. Nesse contexto, o papel das mídias de projeto ganha outro patamar, modifica-se o pensamento por trás do processo de projeto. Com o desenvolvimento das pesquisas, em publicações posteriores, o termo digital é substituído por *Design Computacional* (OXMAN & OXMAN, 2014); (OXMAN, 2017), ampliando-se a abrangência da computação, agora utilizada como forma de auxiliar no desenvolvimento dos projetos.

Figura 2 - Esquema genérico para a representação do processo de Projeto de Design Computacional, proposto por Rivka Oxman.



Fonte: Adaptado de Oxman (2006).

A autora desenvolveu esquemas de representação do processo de projeto (Figura 2), que demonstravam um impacto do *design* computacional, da concepção à construção no processo de projeto. Observa-se no esquema que o projetista [D] passa a assumir a centralidade do processo de projeto, utilizando quatro classes de atividades, já consolidadas na metodologia tradicional, (representação [R], geração [G], avaliação [E] e desempenho [P]). Os quadrados à direita estão associados ao tipo de mídia digital com a qual o arquiteto interagirá. Sendo assim, arquitetura passa a ser vista como resultado da integração do arquiteto com ferramentas computacionais que auxiliam no processo de projeto.

Os modelos de Oxman (2006), embora desenvolvidos há mais de uma década, já abrangiam conceitos relacionados ao *design* algorítmico, ao *design* paramétrico e a metodologias relacionadas a ele, como o BIM. Em estudos mais recentes, Oxman (2017) reitera a importância das mídias digitais e o papel central do arquiteto no processo de projeto, e define novos esquemas representativos da evolução do *design* computacional. Nesse contexto, o arquiteto passa a atuar como criador de ferramentas de personalização da mídia de projeto, devido à crescente sofisticação da mídia de projeto digital e sua capacidade de funcionar de maneira integrada e interativa.

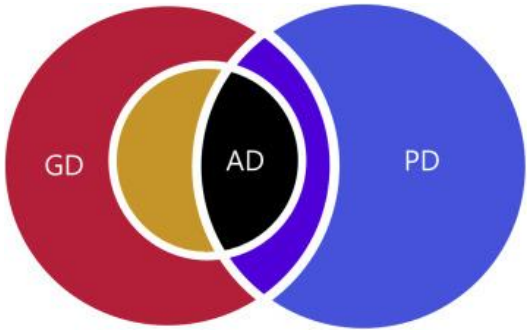
A aplicação do *Design* Computacional na arquitetura está cada vez mais abrangendo novos processos e técnicas, como *scripting*, algoritmos de otimização e fabricação digital (Oxman, 2017), que, por sua vez, originam novos termos relacionados, como *Parametric Design* (Woodbury, 2010), *Algorithmic Design* (Burry, 2011), *Generative Design* (McCormack, Dorin e Innocent, 2004), entre outros.

Com relação ao BIM nesse contexto, Holzer (2015) enfatiza que seus avanços alteram profundamente o papel do arquiteto e engenheiro, pois modificaram a maneira como projetos são pensados, desenvolvidos e compartilhados, aumentando também a responsabilidade e poder de decisão dos profissionais. Para Keough e Hauck (2017), assim como os projetistas do futuro terão um escopo muito maior de alternativas para soluções de projeto, também necessitarão intensificar o uso da computação em seus processos – por meio da otimização, gerenciamento de informações, *design* generativo, algorítmico e adaptativo.

A partir de sua recente expansão, a literatura sobre *Design* Computacional mostra inconsistências nas definições de alguns termos. Isso se dá, principalmente, devido a relações muito próximas entre cada termo e até mesmo a características sobrepostas entre eles. Por exemplo, um tipo de projeto pode ser considerado paramétrico e ao mesmo tempo algorítmico.

Após um levantamento e análises de diversos trabalhos relacionados ao tema, os autores (CAETANO; SANTOS; LEITÃO, 2020) chegaram a uma taxonomia bem estruturada para esses termos, cujas relações estão representadas na Figura 3.

Figura 3 – Relações entre Design Paramétrico (PD), Algorítmico (AD) e Generativo (GD)



Fonte: Caetano, Santos e Leitão (2020).

De acordo com as definições dos autores, nota-se, por exemplo, que todo *design* algorítmico é também generativo. Porém, para ser considerado algorítmico, ele deve obedecer a algumas restrições, como o controle do código. Como exemplo, tem-se um programa que produz um modelo de um edifício criando separadamente suas lajes, colunas, vigas, paredes, janelas e assim por diante. Esse programa é considerado um exemplo de *design* algorítmico porque rastreia as partes do código que produzem uma determinada parte do modelo.

Também é possível que um *design* seja somente paramétrico, como é o caso de uma parede no *Revit*. O usuário pode alterar diferentes parâmetros da parede, como comprimento, espessura e materiais, portanto ela é considerada *design* paramétrico. Porém, como as alterações dos parâmetros não requerem o uso explícito de um algoritmo, não é um exemplo de *design* generativo nem de *design* algorítmico. Outros exemplos dessas relações e suas definições serão melhor abordados adiante.

Soma-se às inconsistências já mencionadas a variação de denominações para traduções de trabalhos para a língua portuguesa. Por exemplo, o termo "*parametric design*" pode ser encontrado em diferentes trabalhos nacionais como "desenho paramétrico" (VASCONSELOS e SPERLING, 2016), "modelagem ou modelo paramétrico" (CÔCO JÚNIOR e CELANI, 2018), (BRÍGITTE, 2019), "projeto paramétrico", "sistemas paramétricos", entre outros.

É importante destacar que, muitas vezes, essas diferenciações são necessárias, de acordo com seus contextos: por exemplo, o conceito de "*design"* paramétrico pode ser considerado mais amplo que o de "modelagem" paramétrica, pois o *design* pode se referir a qualquer projeto desenvolvido a partir de atribuição de parâmetros, enquanto a modelagem costuma ser mais atrelada ao projeto paramétrico desenvolvido no computador.

Porém, não se configura como escopo dessa pesquisa o aprofundamento nas variações de tradução, mas sim nas principais definições de *design* "paramétrico" e "algorítmico", nas suas relações com o BIM, nas características que diferenciam e nas que associam esses termos. Considera-se fundamental para os profissionais de AEC (Arquitetura, Engenharia e Construção) o conhecimento sobre os novos métodos, técnicas e ferramentas com os quais eles passam a interagir através da evolução do *design* computacional.

2.2. *Design* paramétrico

Com base no levantamento teórico mencionado, Caetano, Santos e Leitão (2020) exemplificaram o *design* paramétrico na prática arquitetônica da seguinte maneira:

> Em vez de projetar paredes usando posições, comprimentos, alturas e espessuras exatas, essas propriedades são substituídas por parâmetros simbólicos que têm domínios específicos. O resultado é uma representação simbólica de um conjunto de paredes. Essa abordagem é comumente usada em ferramentas BIM e é expressa no conceito de

> uma família/objeto que descreve conjuntos de elementos de construção. Por exemplo, no caso de uma família de parede, cada combinação de valores de parâmetro corresponde a uma parede diferente. Neste exemplo, existe uma relação direta entre os parâmetros e o *design* resultante, mas em outros casos, essa relação pode não ser evidente porque os parâmetros podem ser usados de uma forma intrincada para produzir *designs* complexos. (CAETANO; SANTOS; LEITÃO, 2020)

Essa definição resume o *design* paramétrico a uma abordagem que descreve um projeto simbolicamente baseado no uso de parâmetros. Para exemplificar, os autores tomaram como exemplo o *design* paramétrico em BIM. Nesse tipo de abordagem existe uma relação direta entre os parâmetros e o *design* resultante.

A definição de Eastman *et al.* (2014) complementa a de Caetano, Santos e Leitão:

> No projeto paramétrico, em vez de projetar uma instância de um elemento de construção como uma parede ou uma porta, um projetista define uma família de modelos ou uma classe de elementos, que é um conjunto de relações e regras para controlar os parâmetros pelos quais as instâncias dos elementos podem ser geradas, mas cada uma irá variar conforme seu contexto. Objetos são definidos usando parâmetros envolvendo distâncias, ângulos e regras como vinculado a, paralelo a e distante de. Essas relações permitem que cada instância de um elemento varie de acordo com os valores de seus parâmetros e suas relações contextuais. (EASTMAN *et al.*, 2014, p. 29).

Nas definições citadas, a modelagem paramétrica não representa objetos com geometrias e propriedades fixas, como seria o caso, por exemplo, de uma parede formada por linhas no *AutoCAD*. Em sistemas BIM, a representação dos objetos ocorre a partir de parâmetros e regras, que determinam a geometria, assim como algumas propriedades e características não geométricas. Os parâmetros e as regras permitem que os objetos se atualizem automaticamente de acordo com o controle do usuário ou mudanças de contexto.

Em uma entrevista (PROCÓPIO *et al.*, 2021) denominada “Parametria e o desejo de uma computação integrada em projeto”, feita com profissionais da área de Projeto, Parametria e Tecnologia de diversas regiões do Brasil, foram reunidas outras conceituações, reflexões e exemplificações sobre o desenho paramétrico e sua utilização no Brasil.

Gabriela Celani reforça o posicionamento de Caetano, Santos e Leitão, enfatizando que:

> Ele [o desenho ou projeto paramétrico] consiste simplesmente em deixar alguns parâmetros do projeto em aberto, especificando intervalos de valores que podem ser atribuídos a cada parâmetro no momento de sua instanciação – ou seja, no momento em que o projeto será efetivamente realizado (PROCÓPIO *et al.*, 2021, p.18).

Nesse sentido, manuais antigos de arquitetura, como os tratados de Alberti, Palladio e Durand, são considerados projetos paramétricos. No texto fundador de *Re AEdificatoria*, por exemplo, Alberti dá grande ênfase aos aspectos formais dos edifícios, principalmente com relação a proporção, definindo-a da forma como a entendemos hoje: “a relação das partes de uma determinada composição entre si e das partes com o conjunto” (ALBERTI apud SOUZA, 2008, p.109).

Já com relação ao projeto paramétrico em computador, Celani (PROCÓPIO *et al.*, 2021) considera que, ao começar a utilização de *software* de desenho, nos anos 1970, já se começava a trabalhar parametricamente.

De acordo com Marcelo Tramontano, a modelagem paramétrica, desenvolvida computacionalmente, tem como diferencial a atribuição de valores numéricos a diversos parâmetros de um projeto de arquitetura. Além disso, esses parâmetros foram associados, de maneira que, quando se altera o valor de um deles, os valores de outros parâmetros também são alterados. Para Tramontano, portanto, o *design* paramétrico em um contexto computacional é basicamente “essa possibilidade de atribuição de valores aos parâmetros e sua manipulação, estabelecendo associações entre eles” (PROCÓPIO *et al.*, 2021, P.23).

Em termos de processo de projeto, Tramontano (PROCÓPIO *et al.*, 2021) enfatiza que o *design* paramétrico altera a maneira como o projetista lida com o projeto: no *AutoCAD*, por exemplo, o arquiteto constrói um desenho através da junção de diversas linhas; já em projetos baseados em linguagem de programação, o programa paramétrico é um interlocutor do projetista, porque se estabelece um processo dialógico justamente por causa da associação de parâmetros (PROCÓPIO *et al.*, 2021).

Portanto, a forma como os parâmetros serão relacionados é fundamental para a definição das geometrias desenvolvidas. A definição de Anja Pratschke (PROCÓPIO *et al.*, 2021) complementa essa visão, pois enfatiza que o *design* paramétrico está diretamente ligado à organização de um processo de projeto. Segundo a autora, a programação visual e o *Grasshopper*, em particular, abriram muitas possibilidades para o desenho paramétrico.

Esse tipo de sistema paramétrico se difere dos sistemas genéricos de representação e modelagem digitais, ao permitir alterar o modelo durante todo o processo de projeto simultaneamente. Permite também gerar e testar grande uma quantidade de versões dentro de um ambiente controlado de projeto. Esses resultados são possíveis a partir da simples mudança de valores de um parâmetro específico.

Jarryer De Martino (PROCÓPIO *et al.*, 2021) considera a substituição do termo "desenho" por "representação" paramétrica. Para ele, o paramétrico está relacionado à criação de um mecanismo capaz de representar a articulação de um pensamento, utilizando-se de diferentes linguagens como forma de registro (matemática, palavras ou símbolos). Já o desenho seria o resultado gerado a partir desse mecanismo de representação. No entanto, embora utilizando-se de outro termo, sua definição de representação paramétrica enfatiza os mesmos pontos que a de Tramontano: "uma forma de representação aberta e dinâmica, registrando a topologia de corpos geométricos e permitindo a sua atualização a qualquer momento" (PROCÓPIO *et al.*, 2021, p. 20).

Embora possam haver diferentes definições para *design* paramétrico devido à amplitude de funcionalidades que essa abordagem abrange, as definições de Caetano, Santos e Leitão (2020) e de Celani (PROCÓPIO *et al.*, 2021) foram as mais implementadas ao longo desse trabalho, visto que não restringem a aplicabilidade do termo *design* paramétrico, ou seja, não colocam determinados requisitos para que um projeto seja considerado *design* paramétrico, como a geometria associativa, a programação ou a fabricação digital. Ou seja, embora esses temas possam se relacionar ao *design* paramétrico, de acordo com esses autores, eles não são condicionantes para sua definição.

Cabe destacar ainda a associação do *design* paramétrico ao algorítmico, recorrente em muitas definições, como a de Oxman (2017, p. 8):

> O *design* paramétrico como um ato de pensamento de *design* é baseado no processo de exploração e reedição de relacionamentos associativos em um espaço de solução geométrica. O designer 'projeta' o código do esquema paramétrico para projetar o objeto de design. O esquema paramétrico é um tipo único de modelo matemático que suporta processos algorítmicos de geração de formas. Assim, embora haja continuidade com as características cognitivas de exploração, geração, reflexão e modificação no design tradicional em papel, a lógica e os componentes sequenciais do processo de *design* foram transformados.

O uso do *Grasshopper* associado ao BIM, que será melhor abordado adiante, consiste em um processo de projeto baseado no *design* paramétrico e algorítmico. É, portanto, fundamental a compreensão do *design* algorítmico antes de se aprofundar nas relações entre esses dois tipos de *design* computacional.

2.3. *Design* algorítmico

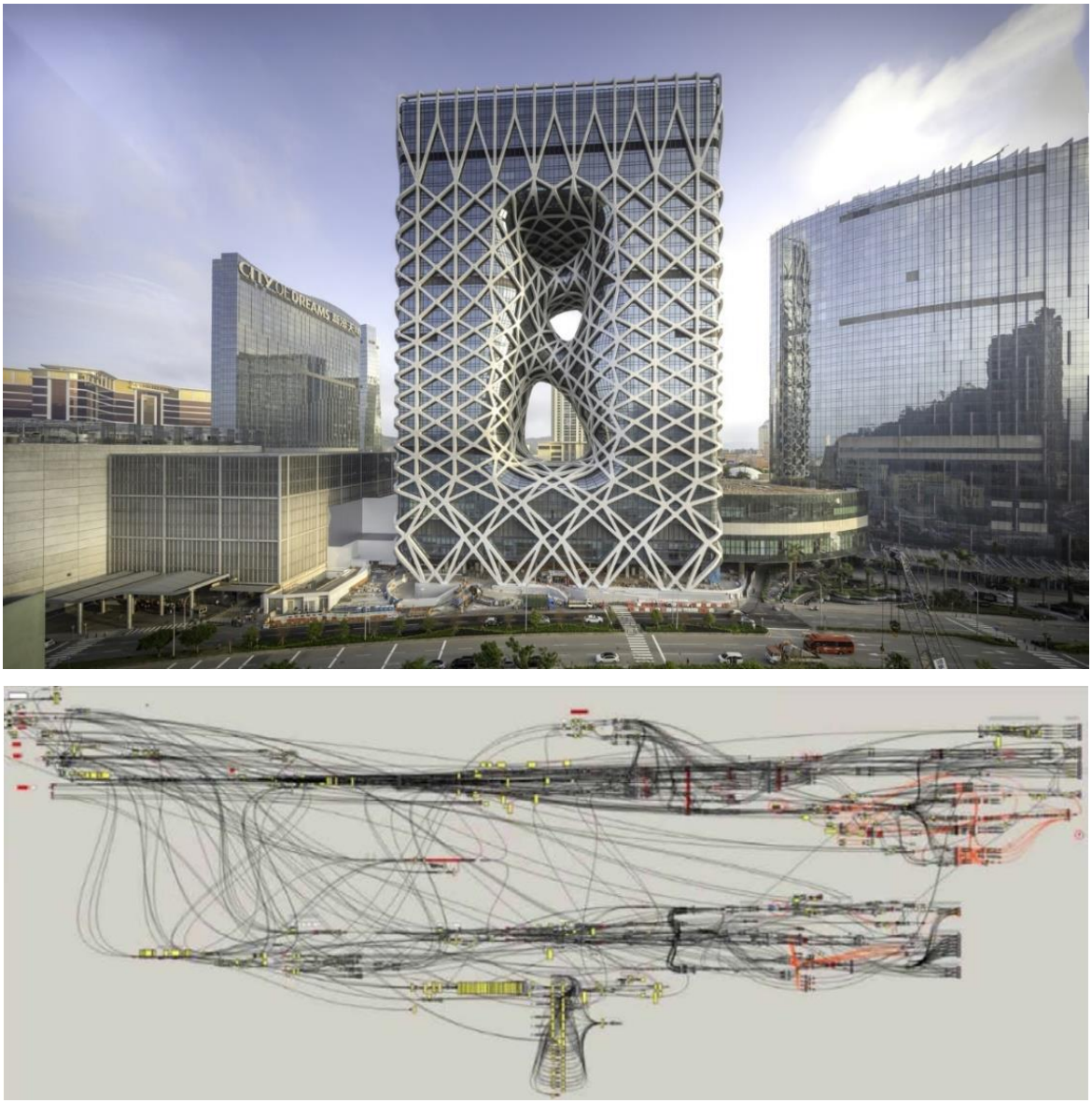
Segundo Oxman (2017), o pensamento algorítmico pode ser definido como um conjunto de regras escritas por um código fonte de instruções explícitas que iniciam procedimentos computacionais que geram formas digitais. Escrever código algorítmico tornou-se um componente fundamental de uma maneira projetual de conhecimento em modelos de *design* algorítmico. Criar *scripts* ou escrever código fornece uma nova maneira de pensar no *design*.

Seguindo com o estudo que visa uma taxonomia melhor estruturada dos termos atuais relacionados ao *design* computacional, Caetano, Santos e Leitão (2020), resumem o conceito de Oxman (2017), chegando à definição de *design* algorítmico como um paradigma de *design* que utiliza algoritmos para gerar modelos. Para os autores, todo *design* algorítmico é também generativo. Porém, eles destacam que o *design* algorítmico possui especificidades: nele, existe uma correlação entre o algoritmo e o modelo gerado, que proporciona rastreabilidade e permite ao usuário identificar as partes do algoritmo que geraram determinada parte do modelo:

> Design algorítmico é um subconjunto de *design* generativo, onde o desenvolvimento algorítmico se concentra no *design* previsto à custa de produzir menos resultados surpreendentes. No entanto, ele fornece um grau de controle mais refinado, facilitando as tarefas de depuração e manutenção (Caetano, Santos e Leitão, 2020, p. 295).

Um exemplo de *design* algorítmico que não seria considerado paramétrico ocorre frequentemente na fabricação digital, quando uma máquina de controle numérico de computador opera e executa um programa que normalmente é gerado automaticamente e de forma não paramétrica.

Figura 4 - *Morpheus Hotel* e o modelo de *Grasshopper* desenvolvido por Zaha Hadid Architects



Fonte: Adaptado de Wortmann e Tuncer (2017).

Aplicando à arquitetura, pode-se citar o *Morpheus Hotel*, de Zaha Hadid, como um exemplo de projeto baseado em *design* algorítmico-paramétrico. O programa de *design* algorítmico que gerou essa edificação é composto por módulos específicos que foram responsáveis por modelar diferentes aspectos da edificação. Na figura 4, tem-se o *Morpheus Hotel* e o modelo desenvolvido no *Grasshopper* para gerar a geometria de revestimento racionalizada do exoesqueleto. Essa forte correlação entre o programa desenvolvido e o modelo gerado é um fator crucial para que o *Morpheus Hotel* seja considerado *design* algorítmico e não somente *design* generativo. E o fato

de seu desenvolvimento formal se basear na atribuição de parâmetros em um *script*, o torna também um exemplo de design paramétrico.

Se, por um lado, o *design* algorítmico possibilita uma grande variedade de formas, análises e estudos do projeto, por outro, seu uso requer conhecimento especializado, o que demanda dos arquitetos conhecimento adicional de outras áreas, como a programação.

A modelagem algorítmica pode ser facilitada a partir de uma linguagem de programação visual (VPL) específica para seu uso. A VPL oferece aos arquitetos que não estejam familiarizados com programação uma interface gráfica que facilita nesse processo, estabelecendo-se graficamente os relacionamentos personalizados.

2.4. Linguagens de Programação Visual (VPL)

De acordo com Leitão e Santos (2011), uma linguagem de programação é um meio formal para traduzir um conceito ou expressar ideias em uma linguagem que o computador entenda e possa executar as operações definidas.
A programação pode facilitar a criação de formas complexas, cuja concepção seria muito difícil ou impossível utilizando apenas funcionalidades básicas das plataformas (AISH, 2013a).

Entre os propósitos mais comuns do uso de linguagens de programação em arquitetura, Leitão (2013) elenca: (a) automatização de tarefas repetitivas (considerado um dos primeiros aspectos usuais da programação geral); (b) obtenção de formas geométricas e funcionalidades que não estão disponíveis nas ferramentas padrão dos *software*; (c) utilização de conceitos de aleatoriedade ou outras regras e variações do mesmo projeto (aspecto fortemente relacionado com o *design* Generativo) e (d) necessidade de que o projeto responda a uma performance/desempenho.

As principais ferramentas para modelagem e desenvolvimento de projetos utilizadas pelos arquitetos se baseiam em sistemas CAD e BIM. A disponibilidade de linguagens de programação nesses *software* se baseia nas Interfaces de Programa de Aplicativos (APIs), que oferecem a possibilidade de trabalhar com programação diretamente no ambiente de trabalho que arquitetos sabem utilizar.

A Tabela 1 apresenta os principais *software* de modelagem CAD e BIM utilizados por arquitetos atualmente. A primeira coluna indica que os *software* estão separados pelos grupos genéricos de modelagem que suportam. No caso desse trabalho,

os grupos estudados foram os baseados em sistemas CAD e BIM. Outros exemplos de grupos baseados em modelagem existentes são os 3D, com *software* como o *Blender*, *Bentley*, *Houdini*, *3D Max*, *Maya*, *Fusion360* e *Unity*; o GIS, com *software* como *QGIS* e *ArcGIS*; e o *Rosetta*, um ambiente de programação que possibilita o uso de diferentes linguagens de programação textual em ambientes CAD e BIM (FERREIRA e LEITÃO, 2015).

Tabela 1 - Principais *software* de modelagem CAD e BIM, e suas linguagens de programação visual e sintaxes disponíveis para desenvolvimento

	Programa/IDE	Linguagem/Biblioteca	Síntaxe
BIM	AUTODESK REVIT	Dynamo	Nodes
		Revit Macros	VB.NET
			C#
		RevitPythonShell	Python
	GRAPHISOFT Archicad	GDL	BASIC-like
	VECTORWORKS	Marionette	Nodes
		Vectorscript	Pascal
		Python	Python
CAD	Rhinoceros	Grasshopper	Nodes
		Python	Python
		Rhinoscript	VBScript
	AUTODESK AUTOCAD	AutoLISP & VisualScheme	LISP
		VB.NET	BASIC-like
	FreeCAD	Python	Python
	SketchUp	Ruby	Ruby

Legenda
Tipo de linguagem
Visual
Textual

Fonte: A autora (2021).

Existem muitas outras nuances de tipos de modelagem dentro desses programas. Por exemplo, os programas CAD *Rhinoceros* e *AutoCAD* se diferem em suas capacidades de modelagem. O primeiro suporta a criação de *NURBS* (*Non-u-niform*

rational basis spline) e *Mesh* (malha), enquanto os sólidos do *AutoCAD* são modelados apenas em *Mesh*[2].

Os retângulos laranja e lilás indicam se a linguagem de programação disponível dentro do programa é visual ou textual. A maioria dos *software* disponibiliza o desenvolvimento de programas através das linguagem de programação textual (TPL), tendo como público principal desenvolvedores de APIs que desejam expandir ou personalizar funcionalidades. No entanto, as linguagens de programação mais utilizadas pelos arquitetos são as de sintaxe "nó" (node), as chamadas VPLs. Esses grupos de *software* com sintaxe VPL são geralmente utilizados quando se referem a Sistemas de *Design* Paramétrico (AISH; HANNA, 2017).

Considera-se Sistemas de *Design* Paramétrico (*Parametric Design Systems*) (AISH, HANNA, 2017) os programas utilizados para o desenvolvimento de projetos paramétricos, algorítmicos e/ou generativos. As VPLs são as sintaxes de linguagem que os sistemas de *design* paramétrico disponibilizam para programação. Essa pesquisa se delineou utilizando o *Rhinoceros* como programa ou sistema que suporta esse tipo de linguagem através da VPL desenvolvida em seu *plugin Grasshopper*. A tabela apresenta outros programas com linguagens/bibliotecas baseadas em VPL, como o *Revit* com o *Dynamo* e o *Vectorworks* com o *Marionette*.

Janssen e Stouffs (2015) distinguem os tipos de modelagem paramétrica com base na maneira em que suportam iteração. Iteração refere-se a mecanismos que repetem sub procedimentos, como funções recursivas, execuções em *loops*, etc. (JANSSEN; LI; MOHANTY, 2016). Segundo os autores, essa classificação é uma maneira clara e inequívoca de classificar os tipos de modelagem, por informar características específicas de cada método e por não agregar a maioria dos métodos dentro da mesma categoria.

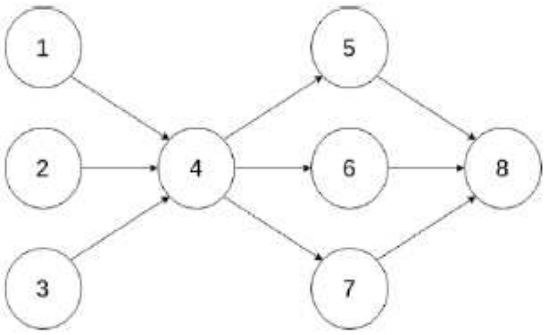
Como descrito na sessão sobre a evolução do *design* computacional, o uso de grafos na arquitetura data da década de 1960 a partir da proposta de Alexander de usar grafos como uma técnica descritiva para modelar a tomada de decisões de projeto (Alexander, 1964). Inicialmente, os grafos ajudaram projetistas a entender a estrutura dos produtos ou sistemas resultantes de seu processo de projeto. Janssen e

[2] Mais informações sobre diferenças nos tipos de modelagem NURBS e MESH estão disponíveis no vídeo do arquiteto Milos Dimcic: ProArchitect #004 - Mesh vs NURBS. Disponível em: www.youtube.com/watch?v=50TeShIuPQU

Stouffs (2015) adotam um Grafo Acíclico Dirigido (*Directed acyclic graph, DAG*) como uma representação genérica para um modelo paramétrico.

Esse grafo consiste em uma rede de nós e arestas que possui uma ordem topológica, uma sequência de vértices, de maneira que cada aresta é direcionada de um vértice a outro e não há possibilidade de retorno de um vértice a ele mesmo. Portanto, esse grafo não pode ter ciclos ou loops.

Figura 5 - Representação de um Grafo Acíclico Dirigido

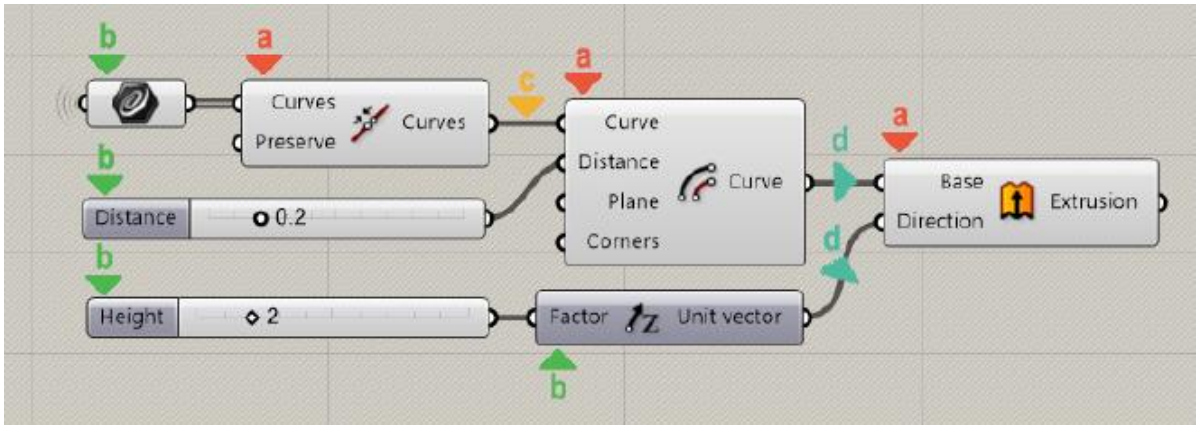


Fonte: Landim (2019).

Na figura 5, os círculos representam os nós, as arestas as conexões e as setas o fluxo de conexão. A partir disso, é possível estabelecer uma relação topológica entre as informações, de maneira que todos os valores das células se ajustem quando um dos valores aplicados aos nós é alterado. De maneira geral, esse é o mesmo comportamento encontrado em um modelo paramétrico usado em arquitetura, por isso a ciência da computação usa esse tipo de grafo para descrever as operações em linguagens de programação de fluxo de dados.

Seguindo essa lógica, Janssen e Stouffs (2015) descrevem um Modelo Paramétrico Generalizado (Figura 6). Nesse modelo, as operações são representadas através da conexão de funções, a partir de “nós” ou “componentes”, que possibilitam a criação de algoritmos baseados em sistemas de regras e restrições parametrizáveis. Portanto, o arquiteto pode tirar proveito das lógicas computacionais, escrevendo um código de programação visual, ou seja, através de “desenhos”, que serão traduzidos automaticamente pelo *Grasshopper* para um conjunto de instruções necessárias para fazer o sistema funcionar.

Figura 6 - Elementos de um Modelo Paramétrico Generalizado, identificados na VPL *Grasshopper*



Fonte: Landim (2019).

É possível observar os seguintes elementos em um código baseado em VPL, no *Grasshopper*:

a. Nós operacionais: a funcionalidade de um nó é determinada pela função implementada em uma linguagem de programação textual. A função do nó é restringida por especificidades do programa base em que opera (por exemplo, *software* CAD e BIM específicos como *Rhinoceros* e *Revit*).

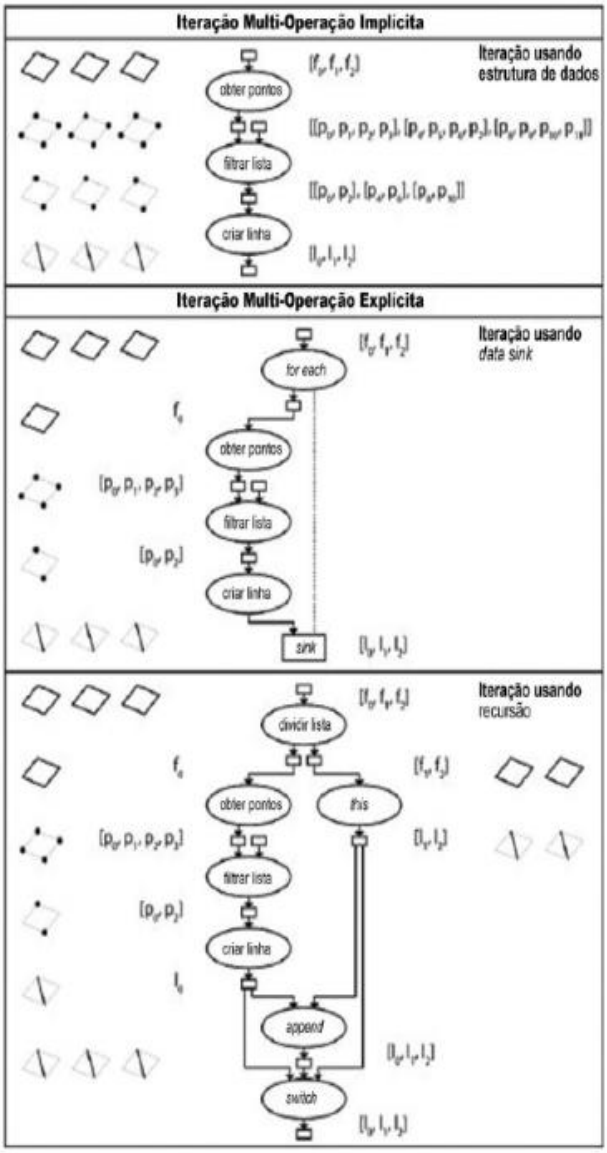
b. Nós de dados: um nó de dados contém um dado ou uma coleção deles, normalmente resultado de uma saída (output) de uma operação ou armazenado para entrar (input) em uma operação (BURNETT *et al.*, 1995). A estrutura desses dados pode variar dependendo do sistema ou software base, mas, de maneira geral, são organizadas como listas planas, listas aninhadas (ou matrizes multidimensionais) e estruturas de dados topológicos (árvores). Em alguns sistemas, os usuários podem manipular a estrutura de dados usando operações ou ferramentas que permitam essa personalização.

c. Arestas ou fios: as arestas representam a direção que o fluxo de dados percorre entre os nós.

d. Execução: a ordem de execução é estabelecida pelo sistema em que o grafo de fluxo de dados está sendo executado. De maneira geral, cada vez que o grafo é executado, os conjuntos de nós operam e resultam em uma saída final. Caso os dados de entrada sejam alterados, o grafo é reexecutado e gera uma nova saída.

e. Iteração (Figura 7): na computação, iteração é o processo de repetição de uma ou mais ações. Em cada repetição (de uma ação), uma ou mais instâncias de iteração podem ser reproduzidas.

Figura 7 - Tipos de iteração permitidos em ambientes de programação VPL



Iteração simples: Nesse tipo de iteração, a mesma operação é aplicada sequencialmente em várias entidades. Exemplo: uma operação geométrica de "extrusão" sobre um nó de dados que contém uma lista de polígonos. Quando executado, o nó de operação com a função extrusão itera sobre uma lista e extruda um polígono por vez.

Iteração multi-operação implícita: a iteração se torna mais complexa quando um nó de operação precisa iterar vários valores de entradas. Exemplo: utiliza-se a mesma operação de extrusão com um nó de dados adicional que fornece uma lista com as distâncias em que cada extrusão deve acontecer. Neste caso, a operação itera sobre ambas as listas, relacionando a posição de cada índice, com o dado da lista correspondente. Neste caso, a maneira como os dados estão estruturados dentro do nó de dados define se a operação irá iterar de maneira correta e resultar em uma saída confiável. O conhecimento de estrutura de dados e tipos de iteração em linguagens de fluxo de dados importa para a saída correta e verificação das informações usadas no projeto de arquitetura.

Iteração multi-operação explícita: segundo os autores, essa operação "representa explicitamente o processo iterativo usando nós adicionais com semântica especializada que modificam o fluxo de controle" (JANSSEN; STOUFFS, 2015, p. 160)

Fonte: Adaptado de Janssen e Stouffs (2015).

Após a identificação dos elementos básicos (a, b, c, d, e) e a divisão de três tipos de iteração suportadas, Janssen e Stouff (2015) dividiram as modelagens utilizadas na arquitetura em quatro categorias, que se distinguem pela maneira que suportam estas iterações (tabela 2).

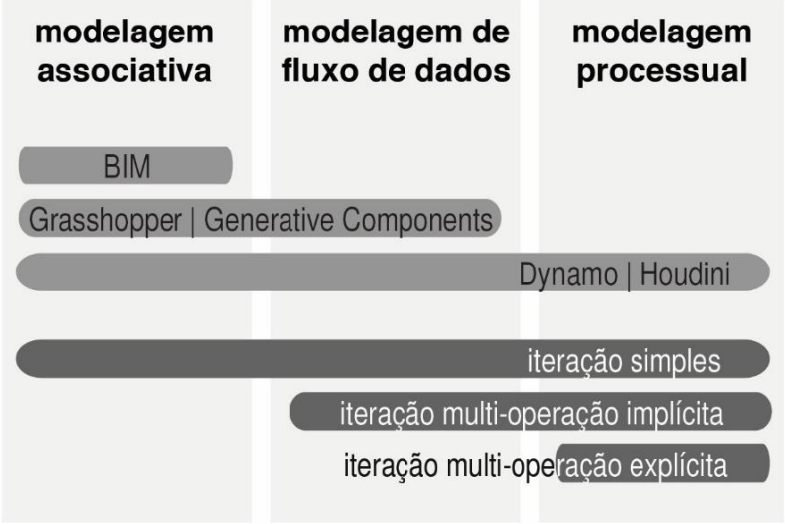
Tabela 2 - Tipos de modelagens e software correspondentes

Tipos de modelagem	Definição	Exemplos
Modelagem de objetos (object modelling)	Categoria de modelagens que **não suportam iteração**. São grupos de geometria com parâmetros definidos.	*P*rogramas de modelagem 3D em geral, como os *"componentes dinâmicos"* do *SketchUp*.
Modelagem associativa (associative modelling)	Suportam **apenas iteração simples** (ou iteração de operação única). Permitem que os objetos modelados possam ser modificados através da mudança de seus parâmetros, sem a necessidade de remodelá-los, uma vez que as famílias e geometrias possuem parametria.	Programas BIM, como *Archicad* e *Revit*. Permitem que os objetos modelados possam ser modificados através da mudança de seus parâmetros, sem a necessidade de remodelá-los, uma vez que as famílias e geometrias possuem parametria.
Modelagem de fluxo de dados (dataflow modelling)	suportam apenas **iteração multi-operação implícita** (ou iteração implícita de múltiplas operações). Podem ser classificados como sistemas que suportam Modelagem de Fluxo de Dados	Interfaces de programação visual *GenerativeComponents* (*Bentley*) e *Grasshopper* (para Rhinoceros, da McNeel)
Modelagem processual (procedural modelling)	suportam apenas **iteração multi-operação explícita** (ou iteração explícita de múltiplas operações).	Interfaces de programação visual *Houdini* (*SideFX*) e *DynamoBIM* (para *Revit* e sua versão *stand-alone* da *Autodesk*)

Fonte: A autora (2021).

Como visto, plataformas diferentes para programação visual em arquitetura suportam tipos de iteração distintos. A partir deste detalhamento sobre as VPLs em funcionamento nos programas paramétricos mais utilizados em arquitetura, é possível compreender que existem diferenças entre os tipos de programação desenvolvidos dentro de cada uma delas. A figura 8 relaciona os tipos de modelagens e iterações que suportam diferentes tipos de plataformas.

Figura 8 - Relações entre os tipos de modelagem e as iterações que suportam



Fonte: Landim (2019).

A busca pela integração entre *software* de modelagem de fluxo de dados (*Grasshopper*) e *software* de modelagem associativa (*Archicad* e *Revit*), como já visto, é extremamente relevante para a evolução do *design* computacional, visto que amplia possibilidades durante processo de projeto: o *Grasshopper* fornece um ambiente de programação visual para definir a lógica algorítmica e gerar o modelo, que, a partir de *plugins* adequados, poderá ser utilizado e editado também em um *software* BIM. Dessa forma, aproveitam-se os potenciais formais e funcionais de ambas as plataformas, sem a necessidade de redesenho ou readequação do projeto a uma plataforma distinta.

De acordo com Silva *et al.* (2019), a programação de extensões VPL, as relações semânticas entre objetos marcam uma das principais diferenças na interatividade quando é feita uma comparação entre softwares de desenho 2D ou 3D simples e softwares BIM. Tratam-se de regras que definem que tipo de objeto interage com outro no modelo e com quais parâmetros. Nos *software* BIM, a programação deixa de manipular apenas fatores geométricos e passa a gerenciar dados de maneira mais ampla (FERREIRA e LEITÃO, 2015). Nestas situações, BIM revolucionou as extensões que utilizam VPL por meio da inclusão desta interdisciplinaridade, possibilitando a criação de uma variedade de programas a serem desenvolvidos na forma de rotinas que manipulem estas relações semânticas (SEGHIER *et al*., 2017).

Embora o *Dynamo* e o *Houdini* suportem diferentes tipos de modelagem, o Houdini é um software 3D, enquanto o Dynamo está embutido em um *software* BIM, o que o torna mais relevante para o uso dentro do Revit. No entanto, comparativamente a ambos, o *Grasshopper* se destaca pela capacidade de funcionar com os comandos nativos do *Rhinoceros,* baseando-se em um conjunto de ferramentas integradas. Diversos *plugins* atuais são interoperáveis somente no *Grasshopper*, elevando a inteligência de modelagem da arquitetura para além do manuseio da geometria. Além de auxiliar na modelagem de formas complexas, é possível associar os modelos gerados ao *Ladybug* para análises de dados climáticos, ao *Honeybee* para análise de modelos energéticos, e a uma série de *Add-os* que expandem ainda mais a capacidade do programa, como o *Anemone*, *Galapagos*, e *Kangaroo*, ferramentas de análise e simulações estruturais, entre outros mais de 300 *plugins* disponíveis no site do programa.

As funções do *Rhinoceros*, expandidas pelo *Grasshopper*, fazem com que essa seja a ferramenta de escolha de muitos arquitetos, designers e engenheiros, por poder controlar e adaptar o modelo de forma interativa, através dos controles e parâmetros, ao invés de ter que desenvolver vários modelos para obter o mesmo resultado. É também a ferramenta escolhida nessa pesquisa, pela capacidade de, após o desenvolvimento formal, interoperar com ambientes BIM a partir de *plugins* específicos.

Alguns *plugins* fazem a conexão do *Grasshopper* ao *Dynamo* para a leitura no *Revit*, como é o caso do *Rhynamo*, que será visto adiante. No entanto, apesar da semelhança de funcionamento do *Grasshopper* e do *Dynamo*, uma conversão manual direta não se resumiria a simplesmente traduzir os nós e a lógica de forma automática. A diferença na maneira como os dados são estruturados no *Grasshopper* e no *Dynamo* dificulta esse processo de interoperabilidade entre eles. Além disso, as entradas e saídas dos componentes (nodes ou nós) são diferentes nessas plataformas, e mesmo os componentes com nome parecidos podem ser diferentes nas funções de entrada e saída. Portanto, para que um trabalho desenvolvido no *Grasshopper* pudesse ser lido no *Dynamo*, e vice-versa, seria necessário reconstruir o gráfico, “nó por nó”, principalmente em grandes códigos visuais.

Apesar das vantagens citadas das VPLs, cabe destacar que alguns tipos de lógicas projetuais são difíceis de expressar e implementar através das VPLs. As linguagens de programação textual podem ser mais abrangentes, mais fáceis de ler e de trabalhar em grandes códigos (LEITÃO e SANTOS, 2013). No entanto, o uso das VPLs na arquitetura permite que geometrias complexas possam ser modeladas sem

conhecimentos detalhados sobre o funcionamento interno do *software*, uma vez que esses ambientes fornecem operações geométricas pré-definidas ao usuário (LANDIM, 2019).

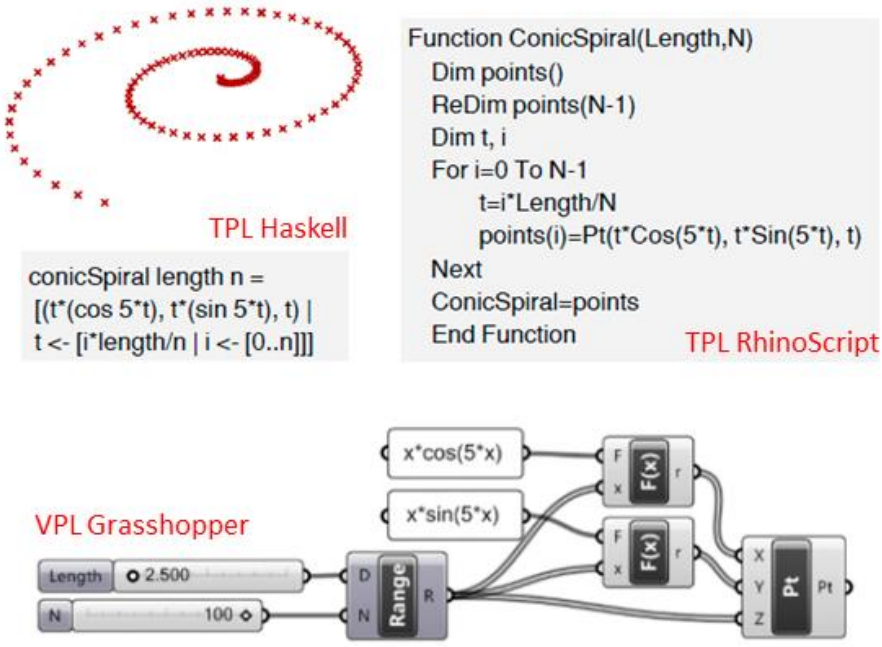
Além disso, conforme apontam Leitão, Cabecinhas e Martins (2010), existem alguns problemas gerais nas linguagens de programação textual dentro das principais ferramentas utilizadas em projetos de arquitetura atuais. Por exemplo: no *ArchiCAD*, a linguagem GDL promove maus hábitos de programação, uma vez que herda diretamente do *BASIC*. No programa *Rhinoceros*, *VBA*, *VBScript* e *RhinoScript* (versões do *Visual Basic*), assim como *Visual Basic* para *Revit* são consideradas linguagens que escondem muitos conceitos importantes de ciência da computação ao custo de serem rápidas para o desenvolvimento de aplicativos. No programa *AutoCAD*, *AutoLISP* (da família *LISP*) é considerada uma linguagem antiga e sem atualizações, oferecendo características obsoletas.

> [...] muitas dessas linguagens de *script* foram desenvolvidas há muito tempo, num momento em que o projeto de linguagem de programação ainda estava em sua infância. A consequência é que as linguagens atuais de *scripting* para CAD não possuem as qualidades pedagógicas necessárias. (LEITÃO; CABECINHAS; MARTINS, 2010, p. 82)

No entanto, também estão disponíveis algumas linguagens de programação textual modernas que avançam os problemas de obsolescência e/ou possuem sintaxe mais fáceis de compreender, como é o caso de: *VisualScheme* (LEITÃO; CABECINHAS; MARTINS, 2010) e C++ para *AutoCAD*; *Python* para *Grasshopper* e C# para *Dynamo*.

A figura 9 ilustra as diferenças de sintaxe entre a linguagem textual *RhinoScript*, a linguagem textual moderna *Haskell* e a VPL no *Grasshopper*. Todos os códigos produzem uma sequência de pontos em uma espiral cônica.

Figura 9 - Diferentes códigos de linguagens de programação de uma sequência de pontos em uma espiral cônica



Fonte: Adaptado de: Landim (2019).

Aish (2013c) argumenta que um sistema que permite a transição “nó para o código” (VPL para TPL), possibilita que arquitetos permaneçam utilizando VPL como ponto de partida e possam migrar para o desenvolvimento de scripts TPL quando for necessário desenvolver lógicas mais complexas. No entanto, o autor também identifica que conceitos adicionais sobre programação serão necessários para programar em TPL.

De acordo com Celani (2012), o raciocínio espacial das VPLs ajuda o desenvolvimento de ideias abstratas, auxiliando o projetista a organizá-las ao planejar uma tarefa que precisa ser programada. Isso acontece porque o processamento de informações visuais em humanos é otimizado para dados multidimensionais, ao contrário de sistemas de visualização unidimensionais (linguagens em forma textual).

Além disso, diversos autores apontam maior facilidade inicial de aprendizagem por parte dos arquitetos das linguagens de programação visual (AISH, 2013c; CELANI, 2012). Outras vantagens são propiciadas pelo *feedback* simultâneo dos resultados de dados e geometrias, com todo o fluxo de operações recalculado e atualizado a cada nova variação de parâmetros (CELANI, 2012).

Portanto, embora tenha-se ponderado sobre as vantagens das linguagens de programação textual, nessa pesquisa objetiva-se o uso de recursos mais próximos dos arquitetos, como a VPL, por ser um tipo de programação de maior assimilação por usuários não programadores. Dentro desse tipo de linguagem de programação, pode-se utilizar, como um complemento aos recursos de modelagem formal do *Grasshopper*, *software* baseados em BIM.

2.5. BIM - Building Information Modeling

Segundo Eastman *et al.* (2014), o BIM é a construção de um modelo virtual preciso de uma edificação, contendo dados relevantes e necessários para dar suporte à construção e incorporando funções necessárias para o ciclo de vida de uma edificação.

> Quando implementado de maneira apropriada, o BIM facilita o processo de projeto e construção mais integrados, que resulta em construções de melhor qualidade com custo e prazo de execução reduzidos (EASTMAN *et al.*, 2014, p. 1).

Montesinos, Galant e Pascual (2018) definiram BIM como uma metodologia colaborativa para modelagem da informação e gerenciamento digital de dados essenciais de um edifício ao longo de seu ciclo de vida. Enquanto nos sistemas CAD a geometria é baseada em coordenadas para o desenvolvimento de entidades gráficas, formando elementos de representação (paredes, portas, lajes, etc.), nos quais qualquer alteração de um projeto desenvolvido em 2D ou 3D implica em diversas modificações “manuais” dos objetos representados, os sistemas BIM adotam modelos paramétricos dos elementos construtivos de uma edificação, permitindo o desenvolvimento de alterações dinâmicas no modelo gráfico, que refletem em todas as pranchas de desenho associadas, bem como nas tabelas de orçamento e especificações.

Essas considerações são explicadas por Eastman *et al.* (2014), ao analisar o processo de projeto BIM, baseado em modelagem paramétrica:

> No projeto paramétrico, em vez de projetar uma instância de um elemento de construção como uma parede ou uma porta, um projetista define uma família de modelos ou uma classe de elementos, que é um conjunto de relações e regras para controlar os parâmetros pelos quais as instâncias dos elementos podem ser geradas, mas cada uma irá variar conforme seu contexto. Objetos são definidos usando parâmetros envolvendo distâncias, ângulos e regras como vinculado a,

> paralelo a e distante de. Essas relações permitem que cada instância de um elemento varie de acordo com os valores de seus parâmetros e suas relações contextuais. As regras ainda podem ser definidas como requisitos que o projeto deve satisfazer, permitindo ao projetista fazer modificações, enquanto as regras verificam e atualizam detalhes para manter o elemento de projeto dentro das regras e avisar ao usuário se essas definições não são alcançadas. (EASTMAN *et al.*, 2014, p. 29).

A base de um sistema BIM é o banco de dados que, além de exibir a geometria dos elementos construtivos em três dimensões e suas correspondentes vistas 2D, armazena seus atributos e, portanto, incorpora mais informação do que modelos CAD tradicionais. Além disso, como os elementos são paramétricos, é possível alterá-los e obter atualizações instantâneas em todo o projeto.

Em outras palavras, um modelador de construção paramétrico BIM permite ao usuário criar restrições, como por exemplo a altura de um nível horizontal, que pode ser vinculada à altura de um conjunto especificado de paredes e ajustada parametricamente, criando um modelo de banco de dados dinâmico vinculado à geometria. Este desenvolvimento atendeu a uma necessidade do setor de AEC, de poder alterar desenhos em várias escalas e em folhas de desenho fragmentadas.

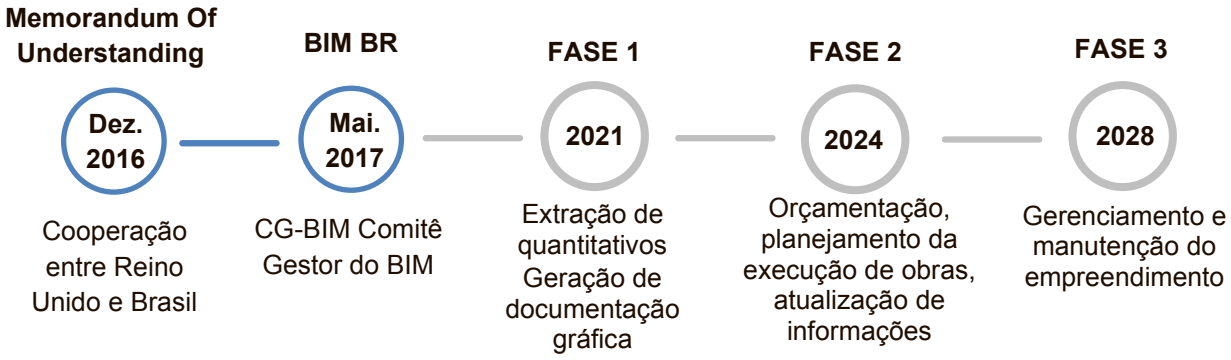
O uso do BIM na AEC traz inúmeros benefícios, tanto para aqueles que participam da cadeia de produção (oferta), quanto para os proprietários e contratantes (demanda): melhoria na eficiência do fluxo da gestão de informações, capacidade de simulação que possibilita antever problemas de custos e desperdícios e melhoria geral da colaboração e da coordenação dos projetos.

De fato, vários países estão encorajando seu uso e adotando medidas para ampliar sua implementação em empreendimentos públicos graças às vantagens econômicas e técnicas que o BIM oferece. Os governos podem desempenhar um papel ativo conectando indústria, tecnologia e educação em todas essas estratégias (LIU, 2015).

A Figura 10 resume as principais iniciativas para implantação do BIM no Brasil ocorridas nos últimos anos. Em dezembro de 2016 foi assinado, no Ministério da Indústria, Comércio Exterior e Serviços (MDIC), um MOU - *Memorandum Of Understanding* (BIMFORUMBRASIL, 2019), que estabeleceu a cooperação entre o Reino Unido e o Brasil para apoiar a iniciativa de desenvolvimento de uma estratégia para implantação e disseminação do BIM no país. Em maio de 2017, no Encontro Nacional da Indústria da Construção, foi assinado o decreto presidencial que instituiu o Programa

BIM BR. Dentre outras ações, o decreto criou o CG-BIM – Comitê Gestor do BIM, de caráter permanente e também sob a liderança do MDIC, tendo como principal incumbência concentrar os esforços para garantir a execução da estratégia de adoção do BIM no país (BIMFORUMBRASIL, 2019).

Figura 10 - Linha do tempo de implantação do BIM no Brasil.



Fonte: A autora (2021).

Em 2019 e em 2020 houveram atualizações no documento, principalmente devido às mudanças ocorridas na criação e fusão de ministérios e secretarias envolvidos que estavam à frente do Comitê Gestor, necessitando que o conteúdo do Decreto fosse atualizado também. O Governo estabeleceu, em 2 de abril de 2020, por meio do Decreto nº 10.306, a utilização do BIM na execução direta ou indireta de obras e serviços de engenharia executados por órgãos da administração pública federal. Foram definidas três fases para a gradativa implementação do BIM, a primeira das quais teve início em janeiro de 2021, a segunda que terá início em janeiro de 2024, e a última, em janeiro de 2028.

Na primeira fase, já em vigor, o BIM deve ser utilizado no desenvolvimento de projetos de arquitetura e engenharia, referentes a construções novas, ampliações ou reabilitações, quando consideradas de grande relevância para a disseminação do BIM, abrangendo: (a) a elaboração dos modelos de arquitetura e dos modelos de engenharia referentes às disciplinas de estruturas; instalações hidráulicas; instalações de aquecimento, ventilação e ar condicionado; e instalações elétricas e (b) a detecção de interferências físicas e funcionais entre as diversas disciplinas e a revisão dos modelos de arquitetura e engenharia, de modo a compatibilizá-los entre si; (c) a extração

de quantitativos; e (d) a geração de documentação gráfica, extraída dos modelos a que se refere este inciso (BRASIL, 2020).

Na segunda fase, o BIM deverá ser utilizado na execução direta ou indireta de projetos de arquitetura e engenharia e na gestão de obras, referentes a construções novas, reformas, ampliações ou reabilitações, quando consideradas de grande relevância para a disseminação do BIM, abrangendo: (a) os usos previstos na primeira fase; (b) a orçamentação, o planejamento e o controle da execução de obras; e (c) a atualização do modelo e de suas informações como construído (*As Built*), para obras cujos projetos de arquitetura e engenharia tenham sido realizados ou executados com aplicação do BIM (BRASIL, 2020).

Por fim, na terceira fase, o BIM deverá ser utilizado no desenvolvimento de projetos de arquitetura e engenharia e na gestão de obras referentes a construções novas, reformas, ampliações e reabilitações, quando consideradas de média ou grande relevância para a disseminação do BIM, e abrangerá: (a) os usos previstos na primeira e na segunda fase; e (b) o gerenciamento e a manutenção do empreendimento após a sua construção, cujos projetos de arquitetura e engenharia e cujas obras tenham sido desenvolvidos ou executados com aplicação do BIM (BRASIL, 2020).

No entanto, para atingir um maior grau de maturidade, algumas questões ainda necessitam de melhorias dentro dos sistemas BIM. Como visto, um dos principais benefícios da modelagem paramétrica é o comportamento de projeto inteligente dos objetos. O problema é que as ferramentas BIM de projeto são inerentemente complexas, pois cada tipo de objeto do sistema tem seu próprio comportamento e associações, e cada tipo de sistema da edificação é composto de objetos que são criados e editados de maneiras diferente (EASTMAN *et al.*, 2014).

Portanto, o tempo de aprendizagem para a proficiência no uso de *software* BIM costuma ser muito maior do que o despendido para *software* de modelagem CAD, como *ScketchUp*, *Rhino* e *FormZ*, que não são ferramentas baseadas em modelagem paramétrica. Essas, têm um modo fixo de editar geometricamente os objetos, variando apenas de acordo com os tipos de superfície usadas. Assim, uma operação de edição aplicada a paredes em *software* CAD, por exemplo, terá o mesmo comportamento quando aplicada a tubulações. Nesses sistemas, caso se queira, o usuário pode acrescentar atributos que definem o tipo do objeto e sua intenção funcional, de forma manual, e não automaticamente ao criar ou acrescentar um objeto previamente configurado, como em sistemas BIM.

A tecnologia BIM ainda não atingiu seu potencial e seu desenvolvimento e implementação acarretam importantes desafios sociais, legais e técnicos para a indústria da construção. Com relação aos desafios técnicos, é importante destacar a necessidade de desenvolvimento de conexões de *software* adequadas entre os participantes do projeto, conceitos abordados dentro da interoperabilidade. Esse canal de comunicação depende, em grande parte, da capacidade de transferir adequadamente as informações entre as áreas de AEC e seus respectivos profissionais.

A partir das discussões apresentadas, entende-se que o BIM deve ser compreendido como uma metodologia, muito além de uma ferramenta de modelagem. Ele representa uma mudança no processo de projeto e gerenciamento. Envolve, além da tecnologia, a integração entre todos os intervenientes do projeto e construção do edifício.

2.6. Interoperabilidade

> Reconhecemos a relevância da integração em aplicativos independentes, mas argumentamos que a natureza diversificada da sociedade e o desenvolvimento de *software* tornam indesejável uma plataforma unificada. Portanto, nossa pesquisa se concentra em aumentar e otimizar a interoperabilidade. (SILVA, Lilian; SILVA, Neander; LACROIX, Igor. 2019, p. 583)

Segundo Kieran, Barry e Zaid (2014), nenhum *software* é capaz de satisfazer as necessidades dos profissionais envolvidos em todo o ciclo de vida de um empreendimento, sendo necessário o uso de diversos programas para a entrega de um projeto. Conseguir realizar a troca de informações com precisão e sem problemas ou perdas significativas de dados, ou seja, conseguir a interoperabilidade entre diferentes *software*, é, portanto, um grande desafio enfrentado pelos projetistas atualmente.

Conforme Eastman *et al.* (2014), a razão para problemas de interoperabilidade entre o *Archicad* e o *Revit*, por exemplo, é que ferramentas BIM diferentes se baseiam em definições diferentes para seus objetos base. Esses são os resultados de diferentes capacidades envolvendo tipos de regras na ferramenta BIM e as regras aplicadas nas definições de famílias de objetos. Tal problema aplica-se somente a objetos paramétricos, não àqueles com propriedades fixas.

A fim de alcançar uma maior integração da informação, os métodos de comunicação vêm se desenvolvendo e usam as APIs de cada um dos modeladores para programar sistemas que transferem a parametrização de um modelo para outro. Cada

solução de modelagem paramétrica usa sua própria linguagem para descrever suas representações. Cada idioma utiliza diferentes termos e sintaxes, que muitas vezes não podem ser traduzidos literalmente para outros sistemas.

Segundo Montesinos, Galant e Pascual (2018), os formatos para troca de arquivos, tradicionalmente usados na indústria de AEC, podem ser dividido em duas categorias: formatos proprietários e não-proprietários. Os formatos proprietários podem ser lidos somente pelo *software* proprietário ou *software* específicos permitidos. Por exemplo, a extensão RVT é perfeitamente lida pelo seu *software* proprietário, o *Revit*. Portanto, o uso dessa extensão pode dificultar a interoperabilidade com outros aplicativos, como, por exemplo, o *software* proprietário *Archicad*, cuja extensão principal é a PLN.

Isso acontece porque a maneira como o *Revit* escreve uma parede simples, por exemplo, e suas relações com o resto dos elementos de um edifício, é muito diferente da descrição do *Archicad*, embora sua aparência externa seja exatamente a mesma. Isso impossibilita a importação de uma representação paramétrica modelada em outro sistema, sem a intervenção de um conversor que atue em ambas as extremidades da transferência (LARRONDO, 2017).

Esse conversor identifica os elementos, parâmetros e relações que podem ser reproduzidos em ambos os sistemas, e garante que em cada um deles um modelo paramétrico que copia as propriedades do outro possa ser gerado automaticamente. Como exemplo, tem-se os *plugins* conectores, como o *GeometryGym, Grevit, Hummingbird, Lyrebird, Rhynamo*, *VisualArq,* entre outros, bem como as ferramentas que conectam bancos de dados bidirecionais de modeladores BIM com tabelas do Excel, que podem ser facilmente editadas, através de outros *software*.

Alguns desses conectores utilizam o IFC (*Industry Foundation Classes*), um formato de arquivo de protocolo internacional (ISO 16739-1:2018) e de código aberto, criado para a troca de informações, atuando tanto no *software* remetente quanto no receptor. É destinado a ser neutro em relação ao fornecedor ou agnóstico e utilizável em uma ampla variedade de dispositivos de *hardware*, plataformas de *software* e interfaces.

Os IFCs são os principais exemplos de formatos de arquivo não proprietários, pois podem ser lidos e editados por qualquer tipo de *software*. Os formatos não-proprietários geralmente são *open source* e o seu desenvolvimento foi fundado na colaboração internacional (MONTESINOS, GALANT E PASCUAL, 2018).

A especificação do esquema IFC é a principal entrega técnica da empresa *buildingSMART International* para cumprir seu objetivo de promover o *openBIM*. Mais especificamente, o esquema IFC é um modelo de dados padronizado que codifica, de forma lógica, as informações de um projeto: a identidade e semântica (nome, identificador exclusivo legível por máquina, tipo de objeto ou função); as características ou atributos (como material, cor e propriedades térmicas); e relacionamentos (incluindo locais, conexões e propriedades) de objetos (como colunas ou lajes); conceitos abstratos (desempenho, custeio); processos (instalação, operações); e pessoas (proprietários, projetistas, empreiteiros, fornecedores, etc.).

Os dados IFC desejados podem ser codificados em vários formatos[3], como XML, STEP e JSON, e transmitidos por serviços da Web, importados/exportados em arquivos ou gerenciados em bancos de dados vinculados. A escolha do formato pode levar em consideração o suporte de *software*, escalabilidade e legibilidade. Como os dados de construção podem ser muito grandes (ou seja, *gigabytes*), a escolha do formato pode ter considerações práticas. O formato físico STEP (SPF ou IFC-SPF) é o mais usado para IFC na prática, e é o mais compacto dos formatos que podem ser lidos como texto. O *Extensible Markup Language* (XML) fornece legibilidade aprimorada e se beneficia de uma ampla variedade de ferramentas de *software*.

Com a difusão da metodologia BIM no *Design* Computacional, espera-se que a troca de dados baseados em IFC possa ocorrer sem problemas entre *software* BIM heterogêneos, como é o caso do *Archicad* e do *Revit*. No entanto, problemas de interoperabilidade ocorrem frequentemente durante trocas de dados bidirecionais usando IFC.

De acordo com Huahui e Deng (2018), as causas fundamentais dos problemas de interoperabilidade via IFC entre diferentes *software* BIM são: (a) ferramentas de *software* não conseguem interpretar bem vários objetos pertencentes a outras disciplinas de projeto devido à diferença no conhecimento do domínio; (b) as ferramentas de *software* têm diversos métodos para representar a mesma geometria, propriedades e relações, levando a dados de modelo inconsistentes.

Já se falou que um dos fundamentos do BIM é a utilização de um modelo único para uma maior segurança e continuidade das informações, evitando redundância no

[3] Mais informações sobre IFC e seus formatos no site oficial da *buildSMART*: https://technical.buildingsmart.org/standards/ifc/ifc-formats/ Acesso em: Nov/2021.

processo. No entanto, Marsico *et al.* (2017) relatam que, ao exportar um modelo em IFC de um *software* para outros desenvolvidos pela mesma empresa, foi possível, em algumas situações, que todos os elementos fossem armazenados com segurança e qualidade. Porém, ao serem importados em outro *software*, alguns elementos, como as paredes sobrepostas em pavimentos diferentes, que não poderiam ser eliminadas, acabaram sendo automaticamente excluídas pelo aplicativo.

No caso do trabalho mencionado (MARSICO *et al.*, 2017), como todos os arquivos eram de um mesmo proprietário (a *Autodesk*), foi possível a importação através de um formato proprietário. No entanto, o intuito era justamente analisar a performance de um formato não-proprietário, o IFC. Nesse caso, a utilização do IFC não foi totalmente eficaz para a troca de dados entre diferentes *software*.

Ricardo Zepeda (2019) considera que, em um projeto onde prevaleça o uso do *Revit* pela maioria dos especialistas, é muito mais fácil integrá-lo a um fluxo de trabalho usando arquivos nativos do que usando o IFC. Segundo o autor, os processos de iteração IFC adicionam uma camada de complexidade e margem de erro no processo de projeto. No entanto, deve ser enfatizado que os formatos IFC são muito valiosos nas etapas de entrega e *As Built*, tendo, portanto, grande relevância dependendo do fluxo de trabalho em BIM.

Eastman *et al.* (2014) afirmam que os problemas de interoperabilidade entre diferentes *software* podem desaparecer se e quando as organizações entrarem em acordo sobre uma padronização para as definições dos objetos. Até lá, intercâmbios para alguns objetos estarão limitados ou falharão. Acredita-se que melhoras virão incrementalmente, à medida que a demanda para resolver esses problemas faça as implementações valerem a pena, e as múltiplas questões relacionadas sejam resolvidas.

Huahui e Deng (2018), realizaram um experimento de interoperabilidade via IFC a partir de objetos padrão e de objetos com forma complexa. Os resultados demonstraram que problemas de interoperabilidade comumente surgem, como o aumento do tamanho do arquivo, tipos de objetos inconsistentes, representação geométrica incorreta, cores diferentes, perda de propriedades e de relações entre as partes do modelo. Constatou-se também que objetos de formas complexas apresentam mais problemas de interoperabilidade, e o maior problema, entre os critérios estabelecidos, se deu na relação entre as partes dos objetos. As principais causas dos problemas detectados foram:

> (1) Existem problemas de interoperabilidade, como perda de dados e representação incorreta, quando ferramentas de *software* importam modelos IFC criados por outras ferramentas de *software*. Essa causa reside nas diferenças semânticas. A ferramenta de *software* específica do domínio pode representar corretamente as informações de seu próprio domínio. Informações de outros domínios, no entanto, podem ser perdidas ou deturpadas pelo *software* devido à falta de conhecimento relacionado ao esquema de dados interno, como tipo de objeto e representação geométrica. (2) Quando diferentes ferramentas de *software* exportam a mesma informação de um objeto, como geometria, propriedades e relações, diferentes entidades IFC podem ser usadas para definir essas informações, levando a diversas representações. A razão para tal diferença são os múltiplos mapeamentos entre esquemas de dados internos de ferramentas de software e esquema IFC; (3) Quando ferramentas de *software* reexportam modelos de dados criados por outras ferramentas de *software*, pode ocorrer aumento nos tamanhos de arquivos físicos, entidades IFC, etc. A causa é que as ferramentas de *software* preferem usar métodos gerais para representar informações indefinidas em esquemas de dados internos, o que irá adicionar mais dados IFC, como, por exemplo, representação geométrica não paramétrica para a forma complexa (HUAHUI E DENG, 2018, p. 553).

Nesse cenário, ter um bom conhecimento dos formatos de arquivo BIM é importante para a seleção de pacotes de *software*, bem como para estabelecer quais protocolos são mais eficientes para trocar informações entre as diferentes áreas envolvidas no desenvolvimento de um projeto.

No caso desse trabalho, um dos intuitos era a utilização de modelos gerados no *Grasshopper* como geometrias nativas no *Archicad* e no *Revit*. Nesse sentido, a definição de interoperabilidade aqui discutida é pautada na realização de troca de informações com precisão e sem problemas ou perdas significativas de dados, não se limitando à utilização do formato de arquivo IFC para a busca a interoperabilidade, devido às limitações expostas.

3. *SOFTWARE*, PLATAFORMAS E *PLUGINS* UTILIZADOS

Neste capítulo são apresentadas as principais ferramentas utilizadas nas investigações e nos testes desenvolvidos ao longo da pesquisa, visando compreender suas lógicas de funcionamento, seus pontos positivos e negativos com relação às necessidades de projeto mencionadas.

3.1. Archicad

O *Archicad*[4] é um *software* de arquitetura BIM desenvolvido pela companhia Húngara *Graphisoft*, em 1982. Ele oferece soluções auxiliadas por computador para lidar com todos os aspectos comuns de arquitetura durante todo o processo de projeto do espaço construído.

Assim como a *Autodesk*, a *Graphisoft* é membro ativa da BuildSMART uma organização internacional que visa melhorar o intercâmbio de informações entre aplicativos de *software* usados na indústria da construção. A empresa foi também uma das fundadoras do conceito *Open BIM*, que suporta a troca de dados BIM entre as diferentes áreas envolvidas em um projeto de AEC, em plataformas *open-source*.

Segundo Andrade (2014), arquiteto e consultor da *Graphisoft* no Brasil, o *Archicad* tem como vantagem seu direcionamento ao público de arquitetos através de uma interface intuitiva e da preocupação em disponibilizar *templates* em conformidade com as normas legais referentes a prática projetual de cada país e demais necessidades locais em termos de bibliotecas e representações.

Com relação aos formatos de arquivos, o *Archicad* pode importar e exportar DWG (*Autocad*), DXF (formato universal), IFC, 3dm (*Rhino*), 3ds (*3D Estudio*), skp (*SketchUp*), entre outros. Ele pode também exportar o modelo 3D e seus correspondente desenhos 2D para o formato *BIMx*, podendo ser vistos em uma série de plataformas móveis ou *desktop* e com visualizadores nativos *BIMx*.

[4] Informações sobre o *Archicad* e sua utilização em: https://www.graphisoft.com/br/archicad/

3.2. Revit

O *Revit*[5] é outro *software* BIM de arquitetura, que contempla também as áreas de urbanismo, engenharia e *design*. Foi desenvolvido em 1997 pela *Charles River Software*. Em 2000, foi lançada sua primeira versão estável. A *Autodesk* adquiriu os direitos do *software* dois anos depois.

O *software* permite aos usuários projetar edifícios, estruturas e seus componentes em 3D e anotações no modelo com desenhos 2D de forma parametrizada. Também possui todas as características BIM levantadas, com acesso de todos os elementos e informações do projeto através de um banco de dados e com ferramentas para planejamento durante os vários estágios no ciclo de vida do edifício, desde o desenvolvimento do projeto até a construção e, posteriormente, manutenção e/ou demolição.

A versão completa do *Revit* (ou *Building Design Suite*) conta com o *Revit Architecture*, voltado ao desenvolvimento de projetos arquitetônicos completos, o *Revit MEP*, que auxilia engenheiros, projetistas e empreiteiros das áreas de mecânica, elétrica e hidráulica e o *Revit Structure*, com ferramentas voltadas aos engenheiros estruturais. Existe ainda o *Revit LT*, uma solução mais econômica e limitada, focada apenas na área de arquitetura e útil para criar projetos de forma isolada.

Além dos principais recursos do *Revit* de modelagens e anotações, pode-se destacar a capacidade de importação e exportação via IFC, o acesso à API, a complementos e a uma ampla biblioteca de conteúdos na *Autodesk App Store* e o *Dynamo*, uma interface de programação visual de código aberto que é instalada com o *Revit* e facilita no processo de modelagem algorítmico-paramétrica de geometrias complexas em BIM.

3.3. Rhinoceros

Desenvolvido por *Robert McNell & Associates*[6], é um *software* CAD proprietário de modelagem tridimensional baseado na tecnologia *NURBS*. Também pode trabalhar

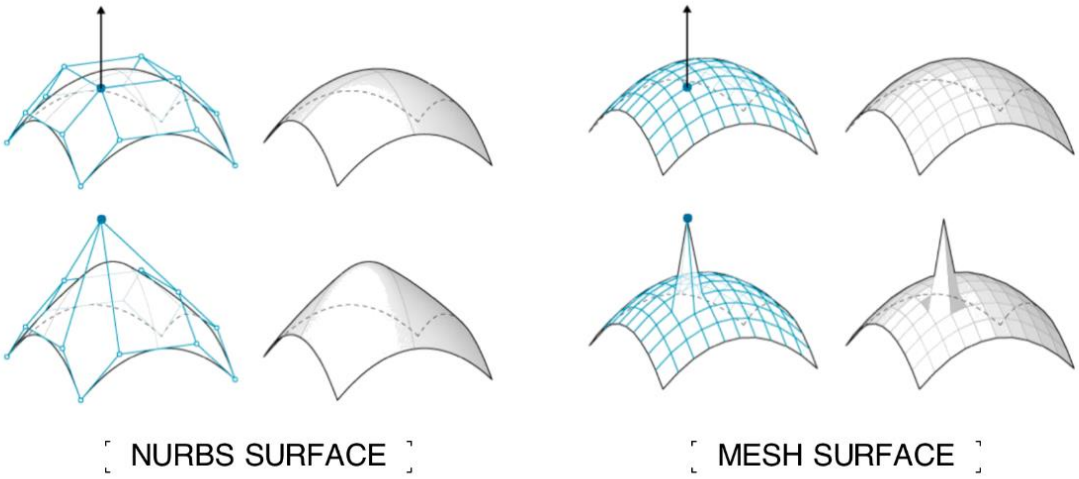
[5] Informações sobre o *Revit* e sua utilização em: https://www.autodesk.com.br/products/revit/overview?term=1-YEAR#workflows

[6] Mais informações sobre o *Rhinoceros* em: https://www.rhino3d.com/ Acesso em: Jun/2021.

com *MESHES* (malhas poligonais). O programa nasceu como um *plugin* para o *AutoCAD*, da *Autodesk*, por isso herda uma série de similitudes com a interface gráfica e barra de comandos. Posteriormente, mais desenvolvido, o projeto se tornou um aplicativo independente. Para compreender o fluxo de trabalho do *Rhinoceros*, faz-se necessária uma maior compreensão dos conceitos de *NURBS* e *MESHES*.

NURBS é um modelo matemático comumente usado em computação gráfica para gerar e representar curvas e superfícies. Esse modelo oferece grande flexibilidade e precisão para lidar com formas analíticas (superfícies definidas por fórmulas matemáticas comuns) e modeladas e, por esse motivo, é tido como padrão da indústria para modelagem e fabricação. Modelagens de alta precisão requerem um grande número de polígonos, e, nesse caso, modelagens em MESH podem se tornar muito grandes.

Figura 11 – Diferenças entre superfícies *NURBS* e *MESH*



Fonte: *Graphisoft Education* (2021).

Na figura 11, pode-se observar que as superfícies NURBS estão sempre enraizadas em uma organização retangular, não são facetadas, e sua descrição permite que a superfície fique lisa, preservando sua qualidade independentemente da escala do modelo. Já as MESHES não possuem restrição de organização retangular, porém sua geometria é facetada, o que pode levar a uma possível perda de qualidade, bem como a grandes tamanhos de arquivos.

Como forma de complementar seus fluxos de trabalho, o *Rhinoceros* traz embutido o *Grasshopper*, um editor de algoritmos gráficos. Como já visto, essa plataforma é usada para projeto algorítmico e paramétrico, desenvolvimento de formas, análise estrutural, análise energética e ambiental, fabricação digital, etc. E também pode ser usada complementando recursos de *software* BIM, com o auxílio de *plugins*.

Uma grande vantagem dos ambientes desse ambiente de programação visual é que ele não requer o aprendizado de uma sintaxe especial. Florio destaca algumas das facilidades trazidas pelo *Grasshopper*:

> Nos últimos anos tem-se acompanhado o crescente interesse dos estudantes de arquitetura por formas e espaços de grande complexidade, em particular edifícios contemporâneos gerados a partir das novas tecnologias digitais. O plugin Grasshopper, com scripts embutidos nos comandos, é o mais popular na atualidade. Criado em 2008, este plugin tem fascinado jovens estudantes a operar com scripts embutidos nas sequências de comandos, que os antigos meios de programação não conseguiram. Sua facilidade de operação tem incentivado a produção de componentes de grande complexidade, com parâmetros claramente definidos. Consequentemente, nota-se que tais recursos tecnológicos têm contribuído para avanços significativos sobre o domínio de formas de grande complexidade. (FLORIO, 2011, p. 45)

3.3.1. Grasshopper

Desenvolvido por David Rutten para Robert McNeel Associates e lançado pela primeira vez em 2007, o *Grasshopper*[7] é um modelador paramétrico de lógica associativa gráfica disponível gratuitamente e um editor de algoritmos integrado à ferramenta de modelagem 3D da *McNeel Rhinoceros*.

O arquiteto ou projetista geralmente abre duas janelas: o espaço de modelagem 3D do *Rhinoceros* e o editor de algoritmos gráficos do *Grasshopper*. Ele pode então se mover entre a modelagem e o ambiente de *script* para criar o *design* paramétrico, a partir de comandos (componentes) e ligações de entrada (*inputs*) e saídas (*outputs*). É possível, por exemplo, criar objetos no *Rhinoceros*, torná-los interdependentes no *Grasshopper* e depois manipular a configuração interdependente no *Rhinoceros*.

[7] Informações sobre o *Grasshopper* e sua utilização em: https://www.grasshopper3d.com/page/scripting-and-code-tutorials Acesso em: Jun/2021.

A imagem do projeto é gerada pelo componente de modelagem *Rhino* 3D do sistema. No entanto, o *Grasshopper* pode se tornar o meio principal e o local do trabalho de projetação formal. O modelo geométrico 3D visível no *Rhinoceros* (controle visual passivo) é conduzido ou executado pelo *script* ativo visualizado e manipulado na janela do *Grasshopper*.

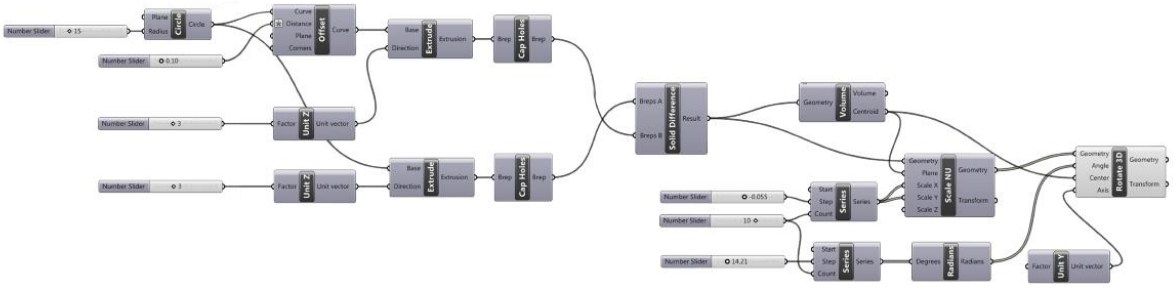
Oxman (2017) se refere ao *Grasshopper* como um aplicativo que funciona baseado em um processo de projeto no qual o usuário “projeta” o código do esquema paramétrico para projetar o objeto de *design*. Esse esquema paramétrico é um tipo único de modelo matemático que suporta processos algorítmicos de geração de formas. Assim, embora haja continuidade com as características cognitivas de exploração, geração, reflexão e modificação no design tradicional em papel, a lógica e os componentes sequenciais do processo de *design* foram transformados.

Com relação à lógica de projeto, Oxman (2017) enfatiza que esboçar por código não é apenas uma possibilidade, mas promete se tornar uma nova norma de habilidade e conhecimento. No *design* algorítmico- paramétrico, o papel tradicional da imagem visual no *design* baseado em papel é substituído pela integração da forma visual (modelo geométrico) e do código visual.

A capacidade paramétrica do *Grasshopper* permite a geração e modificação do projeto simplesmente alterando parâmetros, em vez da necessidade de reescrever quantidades substanciais de código. Esses sistemas possuem um mecanismo de exploração no qual a história da lógica interna de todos os movimentos de modificações pode ser rastreada e refletida.

A seguir, tem-se como exemplo um código visual elaborado no *Grasshopper* (figura 9), de uma luminária pendente baseada na *Moon Lamp* do *designer Verner Panton*. Foi desenvolvido um protótipo em tamanho real (figura 4), e, a partir do código visual criado, tem-se uma variedade de soluções volumétricas (figura 5) geradas instantaneamente com pequenas modificações nos parâmetros e conexões.

Figura 12 - Código visual desenvolvido no Grasshopper de uma luminária parametrizada



Fonte: A autora (2020).

Figura 13 (à esquerda) - Fotografia da luminária desenvolvida no *Grasshopper* à partir do código visual acima e corte na CNC. Baseada na *Moon* Lamp by Verner

Figura 14 (à direita) - Variedade de soluções volumétricas a partir do código visual algorítmico-paramétrico desenvolvido no *Grasshopper*

Fonte: A autora (2020).

Por ser o *Grasshopper* uma ferramenta de código aberto (*open source*), há um estímulo à colaboração da comunidade internacional de usuários, que com seus aportes e novas ideias, compartilham conhecimentos e convertem-se em desenvolvedores do *plugin*.

Como exemplo de *design* algorítmico-paramétrico pode-se citar um programa desenvolvido inteiramente no *Grasshopper*, que produza um modelo de um edifício, criando separadamente suas lajes, colunas, vigas, paredes e cobertura, com possibilidade de rastreamento de cada parte do código para produzir ou modificar cada parte do modelo. Soma-se a isso a transferência de dados dessa modelagem desenvolvida no *Grasshopper* para *software* BIM (*Archicad* e *Revit),* e tem-se o método desenvolvido no quinto capítulo desse trabalho.

Outra possibilidade do Grasshopper é a extensão de suas funcionalidades escrevendo códigos em linguagens de programação textual como VB.NET, C, C#, Python, entre outros (PAYNE; ISSA, 2009). Porém, existem dimensões dentro de um *software* que ainda limitam as funcionalidades que podem ser alcançadas. Ou seja, ao programar um nó em VPL no *Grasshopper*, usando linguagens como VB, C# ou Python, este código poderá sofrer restrição de funcionamento, uma vez que o *Grasshopper* só aceita o tipo de modelagem de fluxo de dados (devido à natureza do Grafo Acíclico Dirigido).

No entanto, o uso das linguagens de programação textual é complexo, sobretudo para arquitetos não-programadores. Por isso, esse trabalho se baseou nas VPLs. Dentro do Grasshopper, diversos plugins também baseados em VPL foram testados, visando conhecer melhor seus recursos de conexão *Grasshoper* x *Archicad*/*Revit*.

3.3.2. Outros plugins para a troca de dados BIM x *VPL*

Para o desenvolvimento do artigo, foram testadas diversas estratégias objetivando a transmissão de dados baseada em *plugins* intermediários entre os *software* BIM e de modelagem algorítmico-paramétrica VPL, com foco na modelagem de geometrias durante a concepção formal do projeto em BIM.

Durante a primeira etapa da pesquisa, foram realizados testes de do *Grasshopper*/*Rhinoceros* com o *Archicad* e com o *Revit,* através de diversos *plugins*. Os resultados alcançados estão descritos no capítulo 5 dessa pesquisa, que dá ênfase a dois *plugins* escolhidos para os testes: *o Archicad Live Connection* e o *Rhino Inside Revit*.

No entanto, considera-se relevante mencionar as características dos outros plugins, de diferentes desenvolvedores, testados ao longo do processo de escolha das melhores ferramentas para solucionar os problemas mencionados.

A tabela 3 descreve as principais características observadas em cada *plugin*. Todos eles possuem em comum, dentre as funções gerais, a capacidade de auxiliar na conversão de geometrias paramétricas desenvolvidas no *Grasshopper* para os principais *software* BIM (*Archicad* e *Revit*), através de diferentes protocolos. O *Dynamo* foi a única ferramenta testada cujo fluxo de trabalho não envolveu a utilização do *Grasshopper*, visto que ele é uma plataforma embutida no *Revit*. Sendo assim, ele sozinho é capaz de gerar geometrias paramétricas lidas e executadas em *software* BIM.

Tabela 3 – Conexões dos principais *plugins* de interoperabilidade entre Grasshopper/Dynamo e Revit/Achicad

Software generativo (protocolo)	Software/ Plugins testados para a conexão	Software BIM (protocolo)	A	B
Dynamo (embutido)	**Dynamo**	Revit (API)		
Rhinoceros (3dm) e Dynamo (embutido)	**Rhynamo**	Revit (API)		
Rhinoceros (3dm) e Dynamo (embutido)	**Mantis Shrimp**	Revit (API)		
GH (API)	**LyreBird**	Revit (API)		
GH (API)	**Hummingbird**	Revit (csv file)		
GH (API)	**Grevit**	Revit (API)		
GH (API e IFC)	**GeometryGym**	Revit (API e IFC)		
Rhinoceros (embutido) e GH (IFC)	**VisualARQ**	Archicad, Revit AECOsim (IFC)		
GH (API)	**Rhino Inside Revit**	Revit (API)		
GH (API)	**Archicad Live Connection**	AC (API)		

Legenda
- Conexão bidirecional
- Conexão unidirecional
- **A:** Conexão em tempo real
- **B:** Open Source
- SIM
- NÃO

Fonte: A autora (2021).

A tabela também traz:

- As interações de cada plugin com outros *software*: alguns fazem a transmissão diretamente do *Grasshopper* para o *Revit* ou para o *Archicad*; outros utilizam o *Dynamo* como mediador entre o *Grasshopper* e o *Revit*, como é o caso do *Rhyamo*, por exemplo.
- Os protocolos utilizados por cada um: o *Dynamo* por exemplo, é embutido no software proprietário (*Revit*); outros, como o *Grevit* utilizam sua API, enquanto o *GeometryGym* pode utilizar IFC para a conexão entre com os *software* BIM. Já o *Hummingbird,* por exemplo, exporta os componentes do GH em um arquivo de texto .csv que pode ser lido pelo *Revit;*
- Se a conexão é bidirecional, ou seja, com possibilidade de alterações na modelagem tanto a partir do *plugin* de modelagem algorítmica no *Grasshopper* ou no *Dynamo* quanto a partir do *software* BIM desejado. As conexões não-bidirecionais possibilitam alterações da geometria somente através do próprio *plugin*, no *Grasshopper*;
- Se a conexão ocorre em tempo real, ou, em outras palavras, se enquanto se alteram os parâmetros do código visual operado, as alterações na geometria são modificadas instantaneamente no *Revit* ou no *Archicad*, ou seja, sem a necessidade de liberação das alterações através de qualquer clique durante o processo;
- Se são *open source*, ou seja, se possuem código aberto, o que possibilita sua livre distribuição entre usuários e desenvolvedores, inclusive de maneira gratuita, podendo ser usados e modificados, adequando-se cada vez mais às necessidades dos usuários.

Na tabela 4, foram levantados os principais pontos positivos e negativos a partir da utilização de cada *plugin* no desenvolvimento de geometrias paramétricas no *Grasshopper* (e no *Dynamo*, como exceção) e na sua conversão para o *Archicad* e para o *Revit*. Alguns plugins testados, como o *Chameleon*, *Open Nurbs* e *Flux.io* não estão mais ativos e, por isso, não entraram na lista. Outros não receberam atualizações nos últimos anos como o *LyreBird*, cuja última atualização se deu em 2014.

Tabela 4 – Pontos positivos e negativos dos principais *plugins* de interoperabilidade entre Grasshopper/Dynamo e Revit/Achicad

Software ou Plugin	**Pontos Negativos**	**Pontos Positivos**
Dynamo	Menos intuitivo e com menos possibilidades que o GH com relação a criação de geometrias	Embutido no Revit; Bom para criação de automações e administração dos modelos BIM
Rhynamo	Necessário o domínio do *Dynamo* e do GH; necessário dar "bake" na geometria para o Rhino	Gratuito; rápida transposição para o Revit
Mantis Shrimp	Interface visual mais complexa, escrito em Python; Necessário domínio do *Dynamo* e do GH	Gratuito; rápida transposição para o *Revit*
LyreBird	Última atualização em 2014; poucos recursos de parâmetros	Gratuito; Interface simples
Hummingbird	Conjunto de componentes limitado	Gratuito; Interface simples
Grevit	Interface menos intuitiva; não recebe atualizações constantemente	Gratuito; suporta envio de geometrias para o *Revit* a partir do GH e *SketchUp*
GeometryGym	Não é gratuito; Limitações de conversão via IFC	Bons recursos para estruturas 3d e análises estruturais; Possibilidade de integração ao RIR para conexão em tempo real
VisualARQ	Não é gratuito; Limitações de conversão via IFC; embora possua conexão com o GH, alguns recursos são somente atribuídos no *Rhinoceros*; alguns comandos possuem lógica de funcionamento diferente da lógica do *Rhino*	Capacidade de edição e adição de *tags* IFC a objetos BIM baseados em NURBS através do *Rhino*; Possibilidade de integração ao RIR para conexão em tempo real

Fonte: A autora (2021).

Tabela 5 – Principais características do *Rhino Inside Revit* e *Archicad Live Connection*

Software **ou** *Plugin*	**Pontos Negativos**	**Pontos Positivos**
Rhino Inside Revit	Demanda um dispêndio computacional maior, o que exige uma máquina com capacidade de processamento mais robusta; Demanda mais estudo e tempo de uso para assimilação das funções e de suas conexões;	Interface mais organizada, a partir de listas conectadas ao *Revit*; Implementação de cores, de acordo com as ações dos componentes; Maior familiaridade com o *software;* Atualizações de rastreamento de elementos
Archicad Live Connection	Interface com menor quantidade de listas de conexão, tornando-a menos intuitiva; Menor quantidade de componentes específicos; Mau funcionamento ou desligamento do *software* às vezes (talvez relacionado a capacidade de processamento do computador utilizado).	Menor tempo de processamento, tornando a conexão mais fluida; Maior facilidade de aprendizagem, devido à menor quantidade de componentes e de possibilidades de conexões de funções;
Pontos em comum	Para utilizar um código visual que tenha sido desenvolvido no GH sem considerar previamente a conversão aos *software* BIM, são necessárias diversas alterações nesse código para a conversão. Requer conhecimento do GH e das lógicas de modelagem e de atribuição de dados dos *software* BIM.	As geometrias são criadas a partir de pontos, curvas ou superfícies, de acordo com a lógica de desenho de cada *software* BIM (uma parede, por exemplo, é desenhada a partir de uma linha no *Revit* e no *Archicad*, e, portanto, será necessária uma linha no GH para a conexão e conversão aos *software* BIM). Possibilidade de complemento de outros *plugins*.

Fonte: A autora (2021).

Dois *plugins* se destacaram, por possibilitarem conexões bidirecionais e em tempo real entre o Grasshopper e o Archicad e o Revit, além de serem open source: o Rhino Inside Revit e o Archicad Live Connection. A tabela 5 relaciona os principais pontos positivos e negativos detectados a partir do uso de ambos os *plugins* para o desenvolvimento de modelagens no *Grasshopper* e sua edição em *software* BIM. A aplicação dos *plugins* será melhor detalhada no capítulo 5.

É importante destacar que a decisão sobre qual utilizar depende, dentre outros fatores, do contexto do projeto, dos *software* aos quais se deseja interoperar, dos protocolos e licenças, do tipo de conexão estabelecida, da necessidade de operação em tempo real ou não e das principais características de cada um.

Para compreender melhor o funcionamento dos dois principais plugins testados durante essa pesquisa, o capítulo 4 traz uma análise de diferentes fluxos de trabalho que associam BIM e modelagem algorítmico-paramétrica.

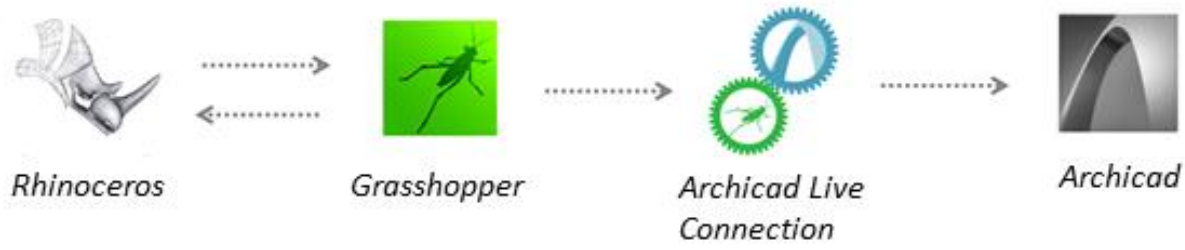
4. FLUXOS DE TRABALHO BIM ALGORÍTMICO-PARAMÉTRICOS

Esse capítulo inicia a aplicação prática das teorias estudadas. Analisam-se os fluxos de trabalho integrados entre *Rhinoceros*, *Grasshopper*, *Archicad* e *Revit*, no intuito de apreender melhor o funcionamento conjunto dos *software* BIM e algorítmico-paramétrico, e também saber quais tipos de fluxos são mais adequados a cada decisão de projeto.

4.1. Fluxos de trabalho integrados entre *Rhinoceros*, *Grasshopper* e *Archicad*

É frequente que se comece com a geometria *Rhino*/*Grasshopper*, para posteriormente convertê-la para o *Archicad* (Figura 15). Nesse processo, o plugin *Archicad Live Connection* preenche a lacuna que existe atualmente entre o *plugin* algorítmico (*Grasshopper*) e o *software* BIM. O contrário - começar com a geometria *Archicad* e levá-la ao *Grasshopper*/*Rhino* - também é possível; as geometrias do *Archicad* podem ser selecionadas como parâmetro de entrada para o *Grasshopper*.

Figura 15 – Exemplo frequente de fluxo de trabalho BIM algorítmico-paramétrico

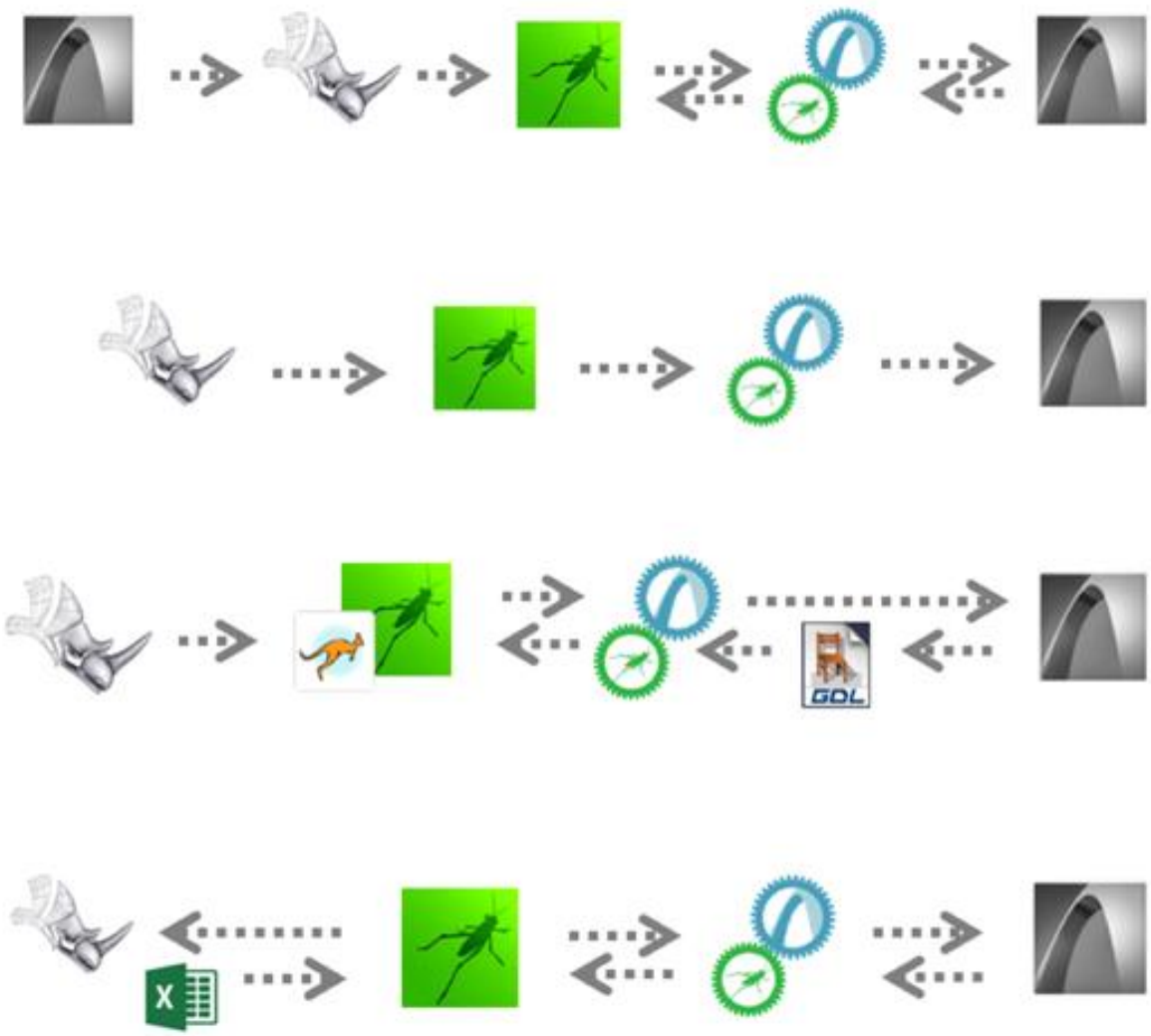


Fonte: Adaptado de *Graphisoft Education* (2021).

Não obstante, diversos outros fluxos são possíveis. A figura 16 ilustra algumas outras possibilidades: as direções dos fluxos estão representadas pelas setas podendo-se, em diversos casos, começar também pelo *Archicad*. No terceiro fluxo, tem-se também exemplos de introdução de arquivos GDL no *Archicad* (esses arquivos podem ser comparados às “famílias” do *Revit*), bem como de *plugins*, como é o caso

do *Kangaroo*[8]. Já incluído a partir da versão 6 do *Rhinoceros*, esse *plugin* auxilia em simulações interativas, desenvolvimento de formas, otimização e resolução de restrições. Outros plugins podem integrar novas funcionalidades ao fluxo de trabalho, dependendo do objetivo do projeto. O quarto fluxo, por exemplo, introduz ao fluxo arquivos csv., que pode ser lido pelo *Archicad*, para a exportação de suas geometrias. Um exemplo de *plugin* que utiliza esse recurso é o *Hummingbird*.

Figura 16 – Outros fluxos de trabalho BIM algorítmico-paramétricos



Fonte: Adaptado de *Graphisoft Education* (2021).

Conforme o primeiro fluxo apresentado (figura 15), foi desenvolvido um algoritmo que cria lajes entre duas superfícies de forma livre no *Rhinoceros*/*Grasshopper*,

[8] Mais informações sobre o Kangaroo em sua plataforma oficial: http://kangaroo3d.com/

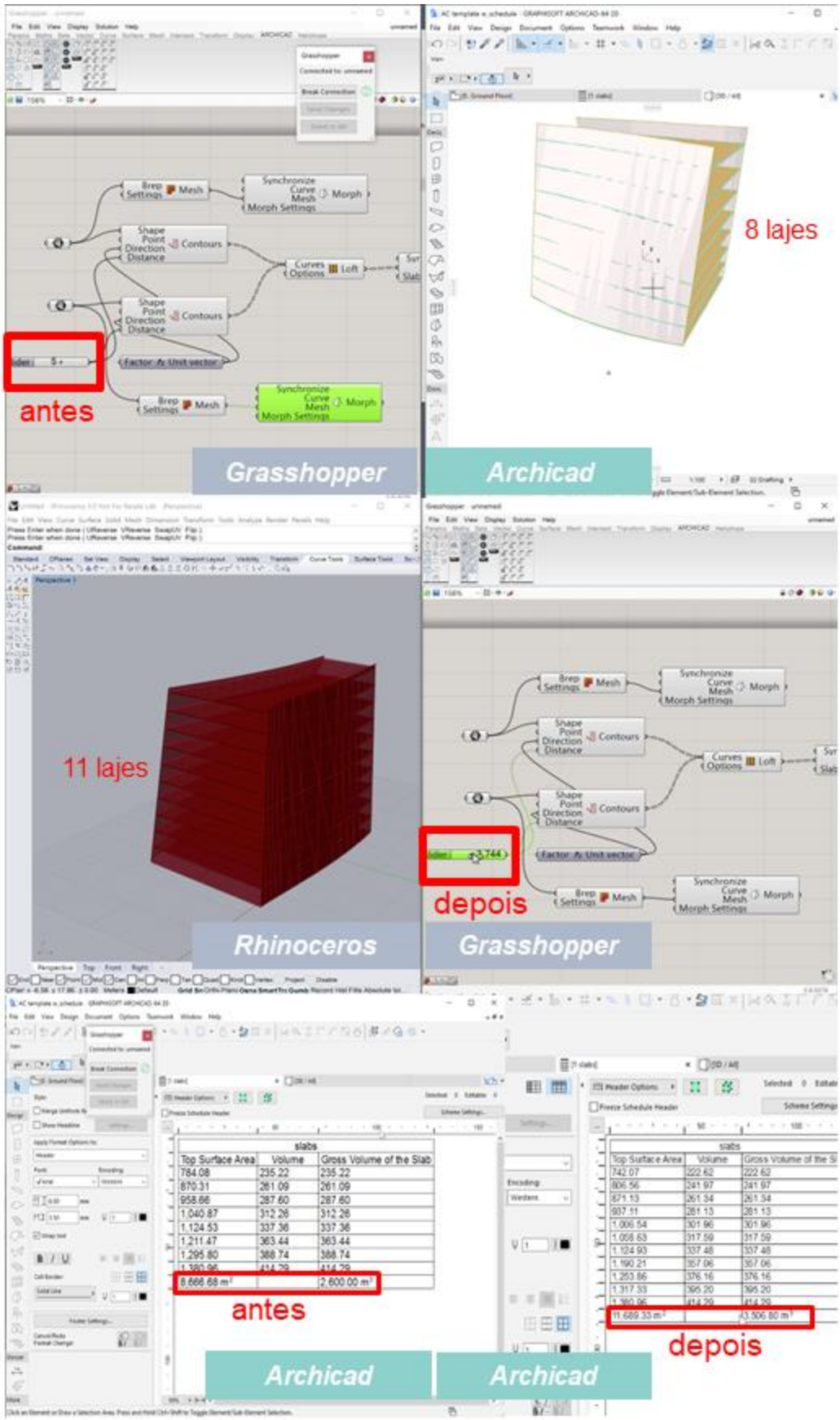
e podem definir as formas de uma torre de vários andares. Resumidamente, esse processo se deu da seguinte forma:

- Foram desenhadas curvas de ponto e superfícies *lofts* no *Rhinoceros*;
- No *Grasshopper*, as superfícies criadas foram referenciadas pelo componente "*surface*", seguido da adição de componentes *Contours* e *Lofts* e seus parâmetros de entradas, para a definição da forma no *Rhino*;
- O componente "*Loft*" é compatível com o componente "*Archicad Slab"*, do *Archicad Live Connection*. Como o nome sugere, esse componente possibilita a utilização da geometria (no caso, das lajes) no *Archicad*;
- Basta abrir o *Archicad* e iniciar a conexão dos *software*;
- Da mesma forma que as lajes, é necessário que as superfícies limitantes do edifício também sejam convertidas em elementos do *Archicad*, para a correta leitura do *software* BIM. Superfícies com curvatura dupla não são conversíveis diretamente em paredes cortina no *Archicad*, mas podem ser convertidas em *Morphs* (ferramenta para formas livres). O componente "*morph"* do *Archicad* usa a "*mesh*" como entrada. Portanto, é necessário converter a superfície *NURB* para *MESH*. O conhecimento sobre as lógicas de modelagem de cada *software* auxilia nesses processos. Para a conversão, tem-se o parâmetro "*Mesh Brep*" no *Grasshopper*;
- Com isso, o componente *"morph"* do *Archicad* Live Connection consegue ler a malha gerada, e as superfícies limitantes são, então, lidas pelo *Archicad*;

A figura 17 resume o processo, ilustrado as interfaces do *Grasshopper*, do *Rhinoceros* e do *Archicad* após a adição dos componentes. Para demonstrar uma possibilidade de alteração do projeto no *Grasshopper* e a repercussão dessa alteração no modelo, foi reduzida a distância entre as lajes do piso, movimentando-se o componente de *number slider* (ver antes e depois no *Grasshopper*). Com isso, o modelo, que antes possuía 8 lajes, passou a ter 11 lajes, para continuar obedecendo às restrições dos parâmetros pré-estabelecidos.

O resultado dessas alterações se reflete não só na forma, como também, por exemplo, em planilhas de volumes e superfícies geradas no *Archicad*. Com o aumento do número de lajes, ambos os valores (de volume e superfície de lajes) também aumentaram e foram automaticamente computados no *Archicad*. Esse pode ser um cenário da vida real, em que requisitos semelhantes são fornecidos e atualizados durante o processo de projeto.

Figura 17 – Utilização do plugin *Archicad Live Connection*



Fonte: A autora (2021).

4.2. Fluxos de trabalho integrados entre *Rhinoceros*, *Grasshopper* e *Revit*

Como já mencionado, o *Rhino.Inside.Revit* permite que formas do *Rhinoceros* sejam codificados e categorizados como elementos do *Revit*. No entanto, a maneira mais fácil e rápida de mover a geometria para o *Revit* nem sempre se traduzirá na melhor estratégia de projeto. Determinar o objetivo final das geometrias no *Revit* pode melhorar a qualidade da estrutura de dados no *software* BIM e aumentar a eficiência do projeto.

O modelo de dados do *Revit* é baseado em um sistema de categorização. Deve-se determinar as melhores categorias e subcategorias a serem utilizadas para que os elementos sejam desenhados e programados adequadamente. Serão abordadas, a seguir, as três principais maneiras de classificar e mover a geometria do *Rhinoceros* para o *Revit*. Cada estratégia sucessiva aumenta a integração dentro de um modelo BIM, mas cada uma também exige um pouco mais de planejamento.

4.2.1. O uso do *DirectShapes*

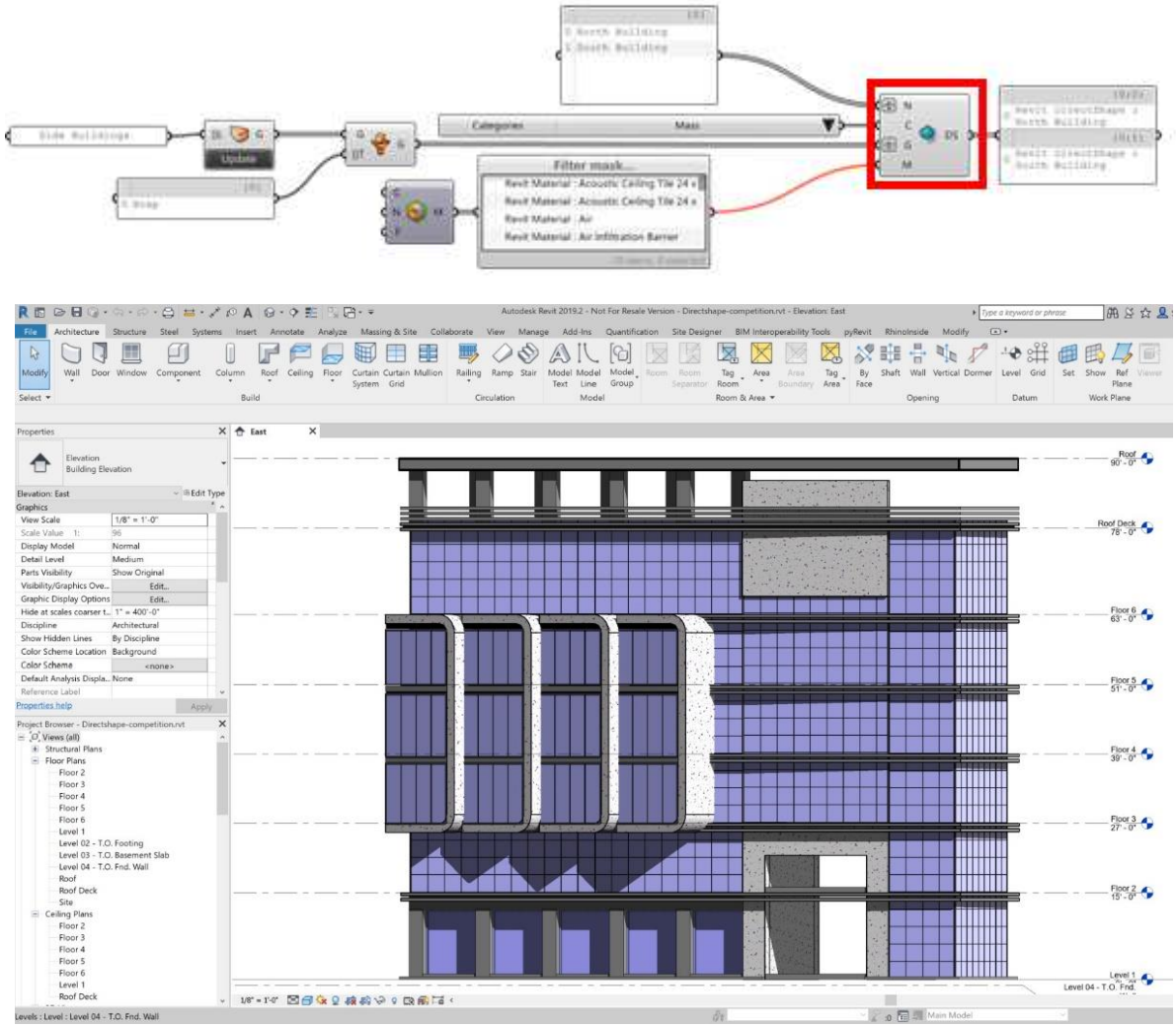
Esse recurso pode ser bastante rápido e requer o mínimo de organização. Porém, a organização e a velocidade limitadas tornam o *DirectShapes* a melhor opção para conjuntos de desenho temporários, como competições e apresentações iniciais de design; no entanto, essa pode não ser a melhor opção para as fases finais do projeto.

Os DirectShapes são elementos genéricos do *Revit* que podem conter e categorizar geometrias não paramétricas arbitrárias dentro do modelo do *Revit*. No entanto, como a geometria não é paramétrica, o *Revit* não sabe como ela é criada, e como consequência, não consegue resolver interações entre *DirectShapes* e outros elementos nativos. Como exemplo, pode-se citar uma geometria de paredes nativas do *Revit* e uma geometria de telhado *DirectShape*. Nessa situação, há um problema de interação e as paredes não podem ser estendidas para alcançar o telhado.
No entanto, em diversas situações o uso do *DirectShape* é vantajoso:
- Em modelos temporários usados em uma competição ou submissão inicial de estudo de projeto para desenhos rápidos.
- Em espaços reservados para parte do edifício que ainda está mudando durante o desenvolvimento do projeto. Por exemplo, enquanto as superfícies de piso podem ser

finalizadas, a fachada pode estar em fluxo no *Grasshopper*. Usar um *DirectShape* como um espaço reservado para elevações e outros desenhos de desenvolvimento de projeto pode ser uma boa solução.

- Em um componente ou montagem totalmente personalizado que não pode ser modelado usando famílias nativas do *Revit*.

Figura 18 – *DirectShapes* no fluxo de trabalho *Grasshopper* x *Revit*



Fonte: Adaptado de *Rhino3d* (2021).

Na figura 18, destaca-se um componente *DirectShape* utilizado para a inserção de uma geometria do *Rhino* no *Revit*. Por meio de um *script* simples do *Grasshopper*, os objetos podem ser categorizados por elevações. Porém, como visto, eles não funcionam como objetos nativos no *Revit*. Neste caso, o modelo foi desenvolvido de

forma rápida para uma competição, não sendo necessário o posterior desenvolvimento do projeto no *Revit*, com atribuições de informações ao modelo, por exemplo.

Os *DirectShapes* criados a partir de superfícies *NURBS* suaves no *Rhino* podem ser importados como sólidos suaves ou convertidos em uma malha pelo *Revit*. Se o *NURBS* for convertido em uma malha, isso pode significar que ela tenha sido rejeitada pelo *Revit*. Dependendo do motivo, esse problema pode ser corrigido no *Rhino*.

4.2.2. Famílias carregáveis com subcategorias

Nesse fluxo de trabalho, os objetos do Rhino são importados como formas dentro de uma família *Revit,* o que permite inserir várias instâncias de um objeto e também atribuir subcategorias. Essas, podem ser usadas para controlar a visibilidade e os gráficos de partes de uma família em uma categoria de nível superior.

Utilizar geometrias do *Rhino* como famílias carregáveis no *Revit* tem como vantagens:

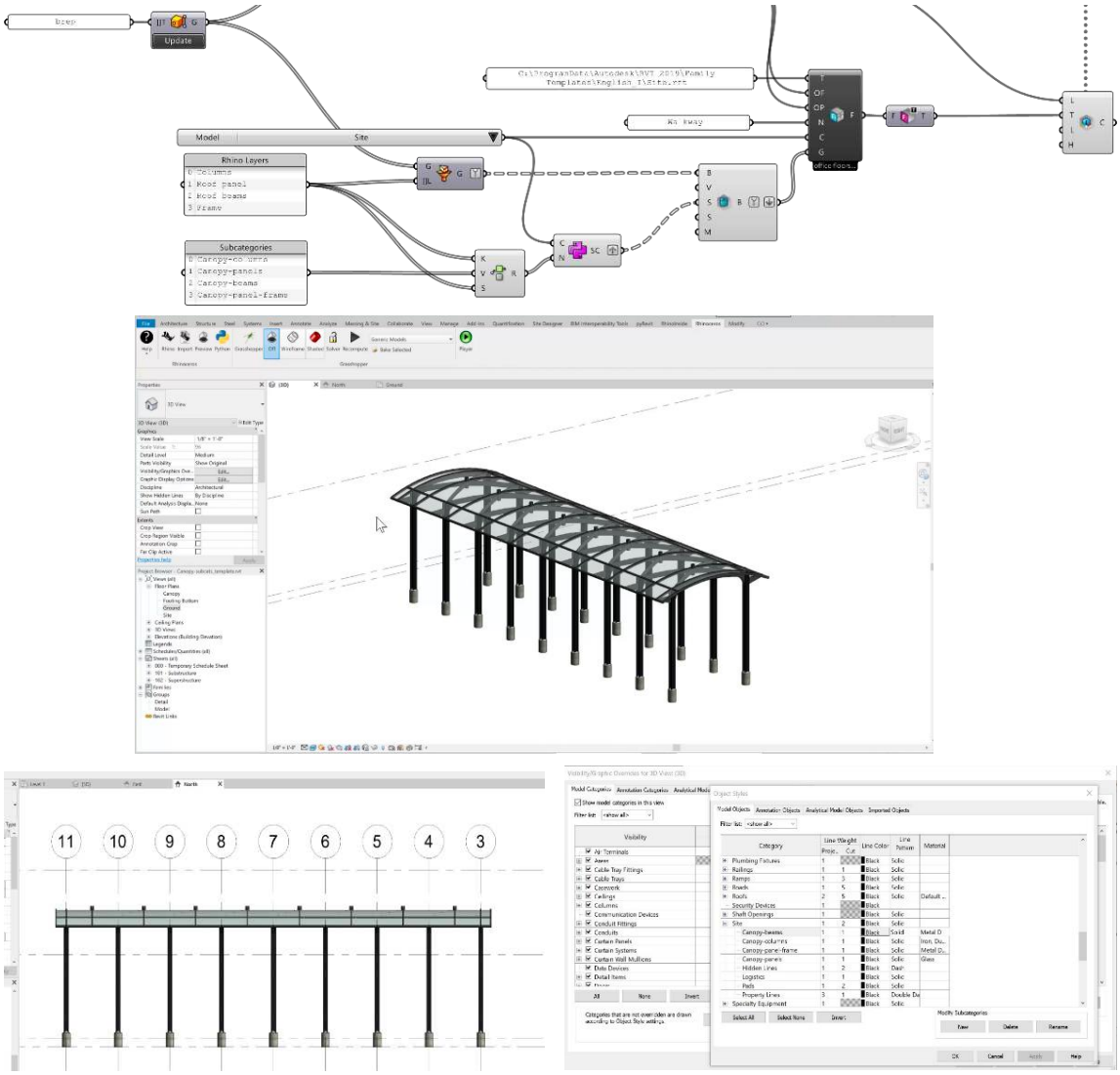
- Objetos repetidos podem ser inseridos várias vezes, permitindo que os formatos sejam identificados e contados corretamente;
- Formas em famílias carregáveis podem ser editados pelo *Revit*, se necessário;
- Formas colocadas dentro de Família/Tipos podem ser colocadas em subcategorias para controle e agendamento de gráficos adicionais.

Esse recurso é útil para elementos independentes em um modelo ou elementos que podem ser encomendados ou construídos por um fabricante independente. Fazendo parte de uma Família, esses objetos poderiam ter seu próprio conjunto de desenhos, além de fazer parte dos desenhos de projetos maiores.

Como exemplo, tem-se uma estrutura de passarela externa desenvolvida no *Rhino/Grasshopper*. Como será construída por um fabricante especializado, as pequenas sapatas serão colocadas no local e o restante da passarela será montado acima. Portanto, as sapatas fazem parte de uma família e o restante da estrutura faz parte de outra família. Ao mapear as camadas do *Rhino* para subcategorias no *Revit*, pode ser utilizada uma tradução automática. Nesse processo, os gráficos e materiais podem ser controlados no *Revit* por subcategoria e visualização. As propriedades da subcategoria podem ser editadas na caixa de diálogo de Estilos de objeto. Observa-

se na figura 19 parte do código visual desenvolvido para esse processo. Para a edição desse código, é necessário um maior conhecimento dos componentes, para que sejam lidos como famílias e atribuídos os parâmetros. Abaixo do código visual tem-se a estrutura no *Revit*, já com os materiais aplicados. Abaixo, tem-se a documentação do projeto, totalmente editável, e uma caixa de edição das propriedades da subcategoria.

Figura 19 – Famílias carregáveis no fluxo de trabalho *Grasshopper* x *Revit*



Fonte: Adaptado de *Rhino3d* (2021).

4.2.3. Modelagem algorítmica *Rhino/Grasshopper* e elementos nativos do *Revit*

Esse é um dos principais recursos empregados nesse trabalho. Como visto, esse fluxo de trabalho funciona de forma semelhante no *Archicad*, porém, com o auxílio de outro *plugin*.

O uso de famílias integradas do sistema *Revit*, como paredes, pisos, tetos e telhados, é mais trabalhoso que os dois fluxos anteriores, e exige certo domínio das ferramentas do *Grasshopper*. No entanto, existem muitas vantagens em se trabalhar com elementos nativos no *Revit*:

- Integração BIM ao processo de projeto, incluindo total controle gráfico da geometria, valores de parâmetros dinâmicos incorporados e acesso a todos os parâmetros BIM padrões de projeto comuns, como qualquer elemento nativo teria;
- Os elementos podem ser editados mesmo quando o *Rhino Inside Revit* não estiver disponível, sem dependência do *plugin*. Eles podem também ter novas configurações no *Revit* e serem usados para hospedar outros elementos;
- Muitos usuários do *Revit* podem não perceber que esses elementos foram criados com o *Rhino Inside Revit*, pois eles se tornam elementos do *Revit*.

Como visto, essa é a melhor maneira de gerar elementos finais no *Revit*. Embora não seja possível criar qualquer geometria com elementos nativos, esse recurso é bastante desejável, pois elementos nativos normalmente se integram melhor com o restante das modelagens dentro do *Revit* e com equipes de trabalho.

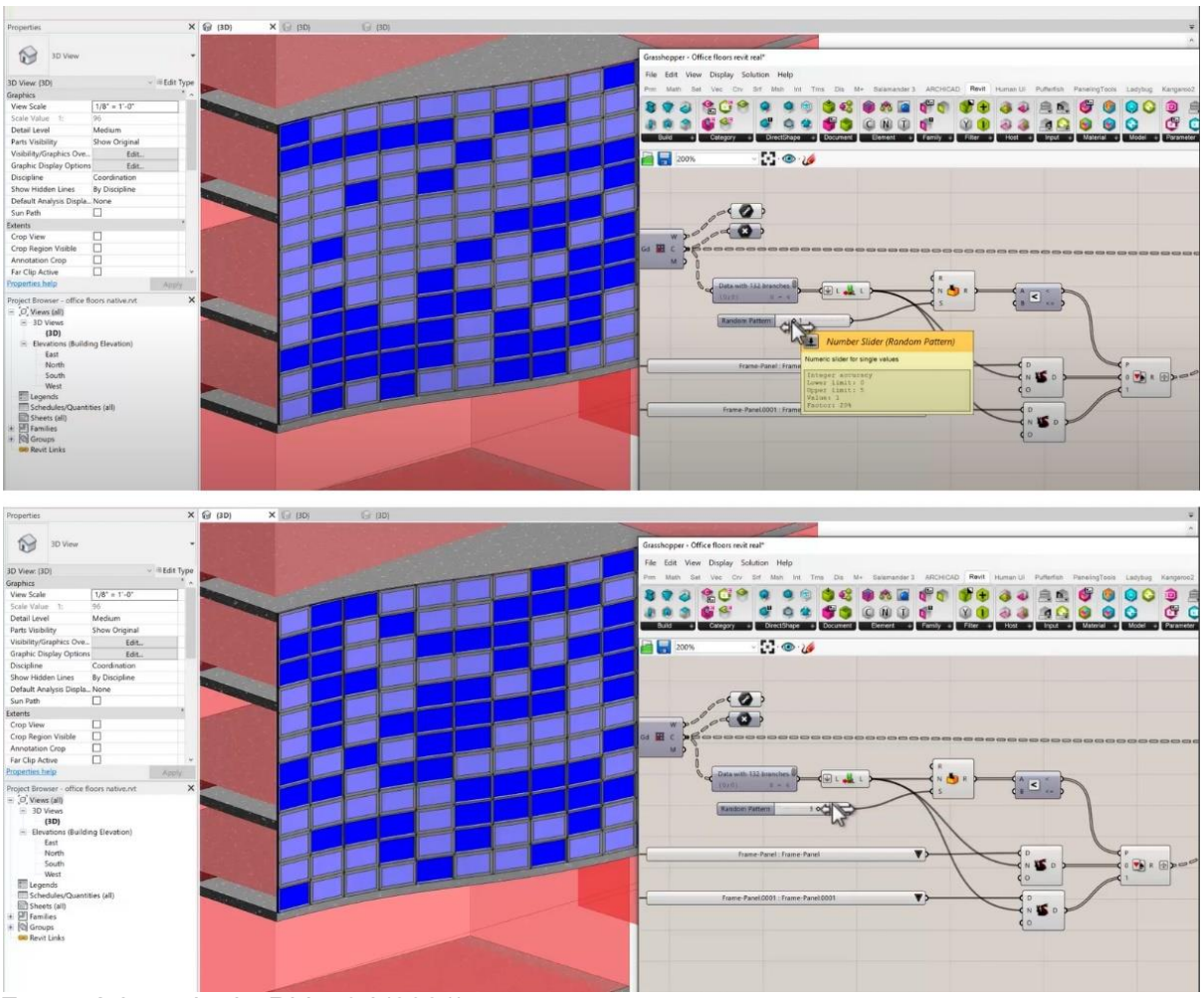
O fluxo de trabalho no *Rhino Inside Revit* é bastante semelhante ao do *Archicad Live Connection*, no sentido de que ambos se utilizam de componentes interligados, estabelecendo-se parâmetros de projeto no *Grasshopper*. E ambos possuem parâmetros específicos para a leitura das geometrias de forma nativa em *software* BIM.

Portanto, para exemplificar o fluxo de trabalho no *Revit*, não foram utilizados os parâmetros tradicionais de paredes, pisos, tetos e lajes; serão utilizados os componentes adaptativos. Esses componentes são uma adaptação do painel de parede cortina com base padronizada. Podem ser usados em famílias de painéis padrão, famílias de componente adaptativo, ambiente de massa conceitual e projetos de *design* complexo. Podem ser usados, por exemplo, em sistemas repetitivos gerados pela ordenação de componentes múltiplos que obedecem às restrições definidas pelo usuário.

Na figura 20, tem-se o ambiente do *Grasshopper* à direita, onde foram inseridos à fachada, aleatoriamente, dois tipos diferentes de componentes de famílias de painéis. Ao “alimentar” o componente adaptativo com essas famílias, o *design* da fachada é criado a partir da combinação das famílias escolhidas, e pode ser modificado de forma randomizada através de um “*number slider*”, gerando diferentes padrões instantaneamente, com possibilidade de modificação rápida. Ao desligar o *Rhino Inside Revit*, os componentes adaptativos formando a fachada envidraçada do modelo

podem ser editados como componentes nativos do *Revit*. Os padrões poderiam ser criados também, por exemplo, a partir de pontos atratores, seguindo a mesma lógica e, com isso, gerando novos padrões de *design* de menor ou maior grau de complexidade.

Figura 20 - Modelagem algorítmica *Rhino/Grasshopper* gerando elementos nativos do *Revit*



Fonte: Adaptado de *Rhino3d* (2021).

Considerações parciais

Com relação aos fluxos de trabalho auxiliados pelo *Rhino Inside Revit*, as possibilidades são múltiplas, assim como no *Archicad Live Connection*. As conexões também podem se dar em ambas as direções (com modelagens feitas no *Rhinoceros*/*Grasshopper* e lidas no *Revit*, ou criadas e lidas no *Revit*, com possibilidade de edições de parâmetros e novas configurações no *Rhino*/*Grasshopper*.

O complemento de outros *plugins* também pode ocorrer nesses fluxos, e, como visto, a amplitude de *plugins* desenvolvidos pela comunidade de usuários do *Grasshopper* é uma das vantagens de seu uso.

Com relação à interface, ao longo do trabalho pode-se observar que, nos últimos 2 anos, a interface do *Rhino Inside Revit* se desenvolveu mais do que a do *Archicad Live Connection*.

Figura 21 – Componentes do Archicad Live Connection e do Rhino Inside Revit no Grasshopper

Fonte: A autora (2021).

O *plugin*, que surgiu no final de 2019 com sua versão *Beta*, *lançou* em 2021 a versão 1.0. Para quem utilizou ambas, não há dúvidas de que a nova versão se tornou mais intuitiva e completa. As categorias possuem bem definidas novas funcionalidades e a divisão dos componentes segue a mesma lógica de suas aplicações no *Revit*. Além disso, foram implementadas cores, de acordo com as ações dos componentes: de consultar, analisar, modificar ou criar elementos.

Para melhorar os problemas de carregamento lento ou "*bugs"* no *Revit*, utiliza-se a visualização alternada nessa interface. O solucionador de alternância também pode ser desabilitado na guia *Rhinoceros, no Revit,* o que auxilia na redução do tempo de espera em grandes modelagens do Revit.

Porém, o ponto de maior relevância a ser mencionado foram as atualizações de rastreamento de elemento. No *Rhino Inside Revit* versão Beta (de testes), elementos do modelo eram duplicados ao fechar e abrir um arquivo. Isso ocorreu, por exemplo, durante os experimentos com a versão *Beta* para o artigo trazido no capítulo 5 dessa esquisa. Com os novos recursos de rastreamento incorporados à versão 1.0, o *Grasshopper* é capaz de substituir os elementos do *Revit* criados anteriormente, mesmo entre os salvamentos. As saídas de componentes passaram a reconhecer quais elementos do *Revit* foram adicionados, o que evita a criação de duplicatas. O

modo de rastreamento pode ser desativado (funcionando com duplicações dos modelos desenvolvidos), ou ativado e gerando novas geometrias, por exemplo, através da substituição por novos elementos ou da atualização dos elementos existentes. A opção de atualização, logicamente, pode deixar o sistema menos lento do que a opção de substituição completa de cada elemento gerado.

Além desses recursos criados, o *Rhino Inside Revit* conta com diversos outros recursos divididos em categorias, famílias, tipos e instâncias. É demandado um pouco mais de estudo e de tempo de uso para a assimilação das funções possíveis, visto que muitas se conectam, gerando ainda mais possibilidades. Além disso, é fundamental a familiaridade com as lógicas de modelagem do *Revit*.

Mas é evidente que os desenvolvedores do *Rhino* têm se dedicado a tentar preencher diversas lacunas de integração BIM x modelagem algorítmica VPL, que estavam em aberto até então. Ademais, a *Autodesk* parece ter se mostrado mais aberta a essas atualizações, visto que o *Revit* (versão 2020)*,* traz como *add-ons* dentro de sua interface a versão 7 do *Rhinoceros* e o *Grasshopper*, através do *Rhino Inside Revit.*

Como aplicação prática das teorias analisadas, no próximo capítulo será realizada a migração de um *script* em *Grasshopper* para o *Archicad* e para o *Revit*, comparando-se os fluxos de trabalho nos dois *software* BIM e levantando-se diretrizes de projeto a partir dos testes desenvolvidos.

5. *INTEGRATION OF BIM AND ALGORITHMIC DESIGN LOGICS THROUGH DATA EXCHANGE BETWEEN GRASSHOPPER PLUGIN AND REVIT AND ARCHICAD SOFTWARE*

Marina Pires Iasbik
Universidade Federal de Viçosa | Brasil | marinaiasbik@gmail.com
Andressa Carmo Pena Martinez
Universidade Federal de Viçosa | Brasil | andressamartinez@gmail.com
Jorge Lira de Toledo Gazel
Universidade de São Paulo | Brasil | jorge.lira@usp.br

5.1. *Abstract*

The Algorithmic Design's integration with BIM (Building Information Modeling), allows greater potential for formal design innovation, tasks automation, greater geometry control, data assignment, and project documentation throughout its life cycle. This paper aims to assist in this integration, analyzing some plugins for conversion from Grasshopper to Archicad and Revit. Based on a parameterized social housing model, interoperability tests were carried out to compare different workflows and discuss some strategies and logics of algorithmic modeling to facilitate the communication between Grasshopper and BIM.

Keywords: Algorithm design. Building information modeling. Parametric modeling. Project process. Interoperability.

5.2. Introdução

Com os rebatimentos de processos digitais de projeto na arquitetura, arquitetos tem buscado conhecimentos cada vez mais diversos para atender às variadas necessidades, adaptações e transformações no campo da arquitetura (FEIST, 2016). Segundo Larrondo (2017), o *design* algorítmico vem se tornando cada vez mais popular nesse cenário, apoiando a criação de formas variadas com relacionamentos parametricamente restritos. Schumacher exemplifica as mudanças positivas na forma de

projetar trazidas pelos auxílios computacionais que constituem o *design* paramétrico e algorítmico:

> O princípio de *design* de “gerar e testar” realizado em um meio ou modelo de design antes da construção física é um substituto econômico (racional) para a “tentativa e erro”, processo que é o princípio da evolução biológica, bem como de toda a evolução cultural pré-arquitetural. Tanto os poderes quanto os fatores da racionalidade do projeto estão sendo massivamente aprimorados pelos auxílios computacionais que constituem o *design* paramétrico e algorítmico em comparação com o *design* tradicional baseado no desenho de acordo com o precedente ou a intuição. (SCHUMACHER, 2016, p.08, tradução nossa)

Juntamente ao *design* algorítmico, o uso do BIM (*Building Information Modeling*) tem se destacado em todo o ciclo de vida de um projeto, permitindo que a documentação bidimensional e a exploração formal tridimensional desenvolvam-se simultaneamente, tornando mais dinâmico esse processo e encurtando o caminho entre o modelo virtual e a representação gráfica (CASTELO BRANCO e LEITÃO, 2017).

Todavia, os *software* mais utilizados para modelagem algorítmica em arquitetura não possuem direcionamento para gerar a representação gráfica e documentação técnica necessárias para a execução destes projetos (GUIDOUX GONZAGA *et al.*, 2018). Portanto, faz-se relevante integrar estas duas atividades projetuais primordiais: a livre criação de formas possibilitada pelo *design* algorítmico e a documentação técnica necessária para execução do projeto arquitetônico, propiciada pelo BIM.

Nesse contexto, esse artigo visa auxiliar na integração das lógicas projetuais algorítmicas e de modelagem da informação para construção, tendo como objetivo principal testar as limitações da transmissão de dados entre um *software* de modelagem algorítmica e um *software* BIM. Para tanto, será feito um levantamento dos principais fluxos de importação de geometrias aos *software Archicad* e *Revit*, através de *plugins* de interoperabilidade, visando gerar um levantamento de estratégias de conversão entre *software* de modelagem algorítmica e BIM.

Esse trabalho constitui a quarta etapa de uma pesquisa sobre customização em massa para habitação de interesse social (GAZEL *et al.*, 2018), que consiste no processo digital das unidades, seguindo as seguintes etapas: (1) pesquisa preliminar de dados de perfil de usuários e modos de morar; (2) a modelagem conceitual arquitetônica; (3) a modelagem algorítmico-paramétrica das unidades no *Grasshopper*; (4)

o desenvolvimento do modelo BIM; (5) definição de plataforma/estratégias para escolhas personalizadas pelos usuários.

Essa quarta etapa consiste no estudo e aplicação dos conceitos de interoperabilidade entre o processo de projeto algorítmico-paramétrico e o BIM. Adicionalmente, também visa explorar a interação dos usuários com os parâmetros modificáveis nos *scripts*, cujas alterações são simultaneamente visualizadas na interface do *software* BIM, tornando o processo também um sistema de suporte às decisões.

Nesse contexto, o conceito de customização em massa de moradias com a plataforma BIM, permite combinar objetivos controversos de individualização e produção econômica, permitindo a máxima flexibilidade no desenvolvimento de projetos com uma estrutura de custos reduzida. Dada a grande quantidade de conteúdo de trabalho personalizado na indústria da construção, a customização em massa surge como uma alternativa eficaz às estratégias atuais de pré-fabricação e padronização (BIANCONI *et al.*, 2019).

Esse artigo, então, apresenta testes de interoperabilidade a partir de um modelo de habitação de interesse social desenvolvido inteiramente no *Grasshopper*, para a transmissão de dados do par *Rhinoceros*-*Grasshopper* para os *software Archicad e Revit*, visando a comparação entre os principais *software* BIM do mercado.

O estudo analisa também o comportamento do modelo no *Archicad* e no *Revit* após a conversão, a partir de testes de modificações bidirecionais e atribuições de dados para a verificação de potencialidades e limitações no processo. Por fim, discute algumas estratégias e lógicas de modelagem algorítmica no *Grasshopper* para facilitar a comunicação com o BIM.

Esse trabalho objetiva, portanto, gerar um levantamento de estratégias atuais de conversão entre *software* de modelagem algorítmica e BIM, bem como definir algumas diretrizes de projeto a partir dos dois *software* escolhidos.

5.3. Materiais e métodos

Este trabalho seguiu as seguintes etapas para a definição do método:
(a) Revisão bibliográfica sobre conceitos e temas como projeto paramétrico e algorítmico, BIM, interoperabilidade e customização em massa na Arquitetura.

(b) Levantamento de *software* e *plugins* para a transmissão de dados entre a plataforma de projeto algorítmico e uma plataforma BIM. Como opção de fluxo de trabalho para a transmissão de dados, para esta pesquisa adotou-se o modelo de importação através de *plugins*. A partir das análises realizadas, dois *plugins* foram eleitos para os testes de interoperabilidade, uma vez que se conectam aos dois *software* BIM mais utilizados atualmente: o *Archicad* e o *Revit*. Deste modo, a conversão foi totalmente executada no *Grasshopper* para o *Archicad* através do *plugin Archicad Live Connection* (ALC) e para o *Revit* através do *plugin Rhino Inside Revit* (RIR).
(c) Testes de interoperabilidade com o modelo de Habitação de Interesse Social (Gazel *et al.*, 2018): Trata-se de um modelo algorítmico-paramétrico de unidades de habitação, com sistema estrutural em aço e peças padronizadas de mercado. Todas as entidades utilizadas na conversão como lajes, vigas, cobertura e paredes estão parametrizadas, o que facilita a conexão com famílias de atributos em ambiente BIM, bem como a transmissão de dados e atualização simultânea do modelo, em caso de modificações nos parâmetros.

Apesar de alguns *software* BIM possuírem nativamente possibilidades de modelagem algorítmico-paramétrica, como o caso do par *Autodesk Revit-Dynamo*, o ambiente *Rhinoceros-Grasshopper* foi adotado por ser um dos mais disseminados na atualidade.

Os *plugins* escolhidos (ALC e RIR) permitem que a geometria seja criada no *Grasshopper* de acordo com a linguagem, a descrição de componentes construtivos e configurações nativas do *Archicad* e do *Revit*, respectivamente. Estes foram os únicos *plugins* testados com conexão em tempo real com os *software* BIM, uma vez que operam bidirecionalmente, mantendo as propriedades intrínsecas do modelo *BIM*, sem a necessidade de parar a exploração formal, exportar a geometria e importar para o novo *software*. Segundo Guidoux Gonzaga *et al.* (2018) este processo representa uma continuidade no fluxo projetual dentro do ambiente digital, tornando-o mais eficiente em comparação aos demais *plugins* analisados, citados adiante.

Além disso, no caso de um projeto de customização em massa de habitação de interesse social, considera-se que a simultaneidade de visualização bidirecional permite aos usuários a possibilidade de modificar os parâmetros associados à geometria, bem como a aplicação de materiais, acabamentos, e famílias que reproduzem peças de mercado. A visualização simultânea do projeto, a documentação gerada (planilhas orçamentárias, quantitativos), dentre outras vantagens, permitem

diretamente a aplicação de conceitos de customização em massa na Arquitetura, Engenharias e Construção.

5.4. Fluxos de trabalho

Visando adequar-se às tecnologias BIM, muitos escritórios de arquitetura têm implementado o uso do *Archicad*, do *Revit* ou do *Vectorworks*, sobretudo para documentação dos projetos. No entanto, para o desenvolvimento de geometrias complexas, destacam-se as ferramentas de modelagem algorítmica como *Grasshopper* (GH), *Blender* ou *SideFX Houdini*. Essas ferramentas de modelagem de fluxo de dados e procedimentos suportam a iteração de várias operações, possibilitando a criação rápida de geometrias complexas, enquanto os sistemas BIM são mais viáveis durante as fases posteriores, de desenvolvimento do projeto e documentação.

Kaushik (2017), argumenta que uma das principais barreiras à integração com o BIM desde o início do projeto é a divisão dos contratos em duas etapas sequenciais e não simultâneas: o projeto conceitual e o projeto detalhado. Após a concepção, normalmente outra equipe assume o controle (ou a mesma equipe), com o modelo, no entanto, reconstruído no *Revit* ou no *Archicad*. Nesse processo, muita inteligência paramétrica e associativa incorporada ao modelo conceitual é perdida ou desperdiçada para obedecer às limitações da ferramenta BIM.

Portanto, visando a integração em um mesmo projeto de sistemas algorítmicos e BIM, é importante considerar alguns fluxos de trabalho diferentes para a conversão de geometrias criadas no *Rhinoceros*-GH em um *software* BIM, como o *Archicad* ou o *Revit*. Cada tipo de importação possui suas vantagens e limitações:

Um dos fluxos de trabalho mais simples é a exportação de um arquivo .sat ou .dwg para o *Revit,* por meio de uma família de massa conceitual. No entanto, a geometria resultante dessa importação não pode ser editada, diferentemente dos elementos nativos do *software*.

Um segundo fluxo de trabalho é a importação por face. Nessa abordagem, é possível a criação de elementos nativos do *software* BIM, como o *Revit*, a partir de superfícies de elementos importados do *Rhinoceros*, utilizando-se os comandos “parede por face”, “telhado por face”, “pisos por face” e “sistema cortina por face”. Esses elementos são hospedados na geometria *Rhino* importada e podem ser atualizados

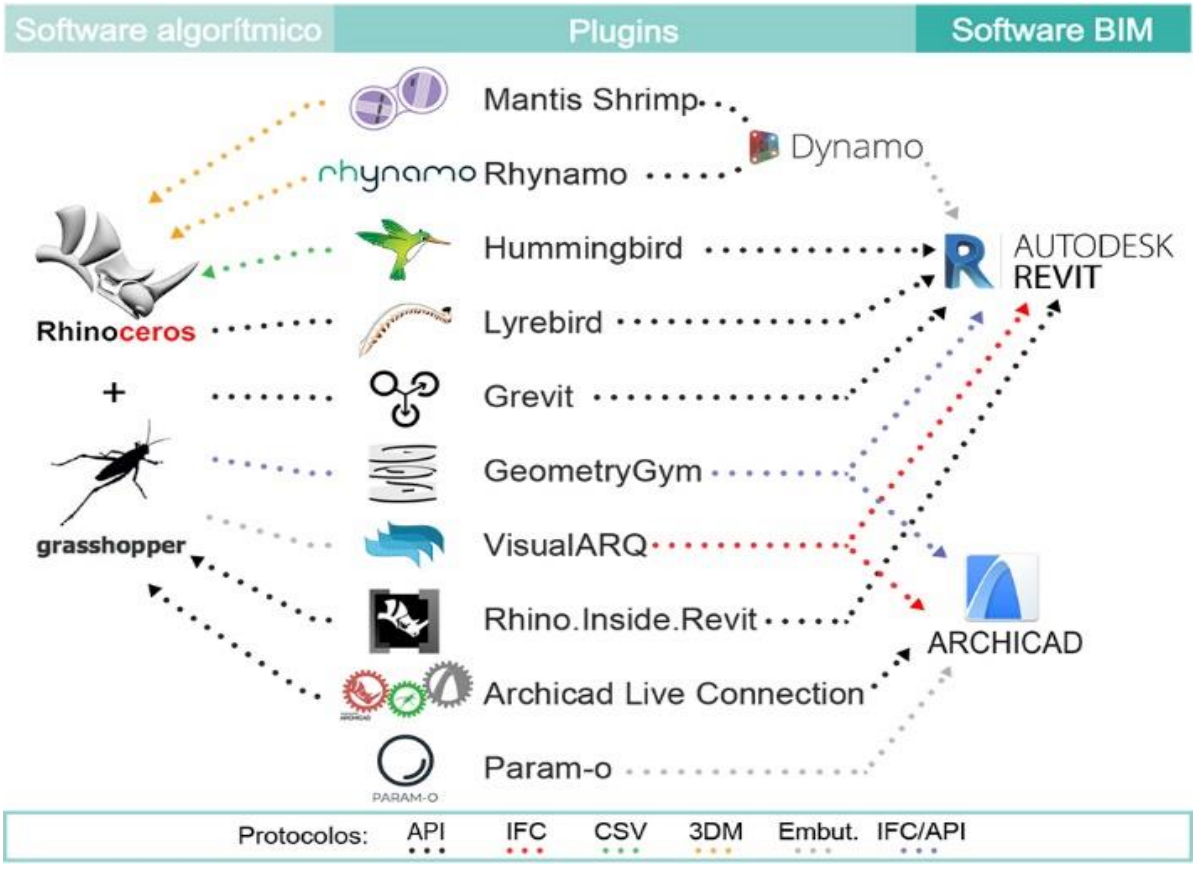
se a geometria base for alterada. No entanto, nem sempre o *software* BIM reconhece ou redefine o elemento *host,* sendo necessária a criação de novos elementos.

Um terceiro fluxo analisado, e com resultados mais promissores, são as conversões auxiliadas por *plugins*, melhor detalhadas a seguir.

5.4.1. Conversão através de *plugins*

Os *plugins* de interoperabilidade para a conversão de dados de um *software* de modelagem algorítmica para o BIM propiciam melhores fluxos de trabalho, seja na criação de geometrias complexas, seja na iteração de várias operações para um controle mais preciso dos elementos BIM e suas propriedades. A figura 22 ilustra os protocolos e conexões uni ou bidirecionais (representadas pelas setas) dos *plugins* testados na pesquisa, para a transmissão de dados do *software Rhinoceros* com as operações algorítmicas do *Grasshopper* aos *software* BIM *Revit* e *Archicad*.

Figura 22 – Plugins Testados e seus fluxos de trabalho na transmissão de dados do *Rhinoceros* + *Grasshopper* ao *Revit* e ao *Archicad*



Fonte: Autores (2020).

Janssen, Chen e Mohanty (2016) dividem esses fluxos de trabalho, diferenciando as suas relações de acoplamento em sistemas *Tightly Coupled* (fortemente acoplados) e *Loosely Coupled* (fracamente acoplados).

5.4.1.1. *Tightly Coupled*

Com a “abordagem fortemente acoplada”, os sistemas são acoplados por meio da API (*Application Programming Interface*) fornecida pelo sistema BIM. Nesse caso, os sistemas baseados em grafos comunicam-se instanciando diretamente a geometria no modelo BIM cada vez que o modelo é executado (JANSSEN *et al.*, 2016). Alguns exemplos são o *Autodesk Dynamo*, em conexão direta com o *Revit, que* utiliza sua API e o *Bentley GenerativeComponents*, que também utiliza a API, com conexão embutida com o *AecoSIM* e o *Marionette*, do *Vectorwork*. Eles são voltados principalmente para usuários que preferem permanecer no *software* BIM, com o qual estão familiarizados.

Recentemente, a *Graphisoft* incluiu o **Param-o** para a criação de objetos paramétricos no *Archicad* 24. Para a criação de objetos GDL, o *plugin* é intuitivo e eficiente, sobretudo quando já se tem noção da lógica projetual do *Grasshopper*. No entanto, para o desenvolvimento, como é o caso da pesquisa, de uma planta parametrizada, o *plugin* é limitado, uma vez que sua lógica é voltada à criação de objetos separadamente para a biblioteca e aplicação no *Archicad*, e não para criação de toda uma estrutura construtiva parametrizada de maneira integrada entre seus módulos, pilares, vigas, lajes e fechamentos.

Kaushik (2017) apresenta algumas limitações desses formatos e *plugins* embutidos nos *software* BIM: como os modelos BIM são, por sua própria natureza, grandes conjuntos de dados complexos, permitir que os usuários explorem parametricamente esses modelos pode reduzir drasticamente a latência e a robustez do sistema. Alterações paramétricas em modelos grandes podem tornar o *software* lento e muitas vezes resultar em erros inesperados, mesmo para usuários experientes. Além disso, os sistemas BIM já possuem interfaces de usuário muito complexas e a adição de recursos avançados de fluxo de dados e modelagem de procedimentos pode resultar em uma interface excessivamente complexa para ambos os casos de uso (KAUSHIK, 2017).

Nesse sentido, muitas ferramentas de interoperabilidade foram desenvolvidas para melhorar as transmissões de dados, visando a integração da geometria *Rhino-GH* com BIM. O GH possui *plugins* para conexão tanto ao *Archicad* (*Archicad Live Connection, Rhino GDL Converter*) como ao *Revit* (*Grevit, Hummingbird, Lyrebird, Rhino Inside Revit*). Existem também *plugins* que convertem geometrias *Rhino* para *Dynamo* (embutido no *Revit)*, como é o caso do *Rynamo*, e do GH para *Dynamo* através do *Mantis Shrimp*.

O ***Hummingbird***, por exemplo, exporta os componentes do GH em um arquivo de texto .csv que pode ser lido pelo *Revit*, para a exportação de suas geometrias. O arquivo de texto pode ser visualizado no *Hummingbird CSV-Viewer* (ou *Excel*) para estudo e edição de dados, caso necessário. Ao instalar esse *plugin*, seus componentes aparecem na guia "extra" do GH e na guia "*Add-ins*" do *Revit*, com o suplemento "*H Bird*". Esse *plugin* não visualiza a geometria no *Revit*, mas reconecta todos os dados necessários para, ao criar o arquivo .csv, reescrever o código, importando-o no *software*. Esse *plugin* possui conexão bidirecional com o *Revit*, é gratuito e de código aberto. No entanto, não possui conexão simultânea com o *Revit*.

O ***Lyrebird*** opera de modo semelhante ao *Hummingbird*, porém instanciando as geometrias através da API do *Revit*. Sua interface é muito simplificada, com apenas um componente "*LBOut*" no GH, que possui *inputs* (dados de entrada) e *outputs* (dados de saída) com diversas funções. No *Revit*, há quatro comandos adicionados para gerenciar o *Lyrebird* e os elementos que ele cria. Esse *plugin* é gratuito e de código aberto. No entanto, não é bidirecional e não possui conexão simultânea. Além disso, não recebe atualizações desde 2014, o que o torna limitado em relação às correções de erros e às mudanças nos demais *plugins*.

O ***Grevit*** também converte as geometrias através da API do *Revit*, operando unidirecionalmente, de forma semelhante ao *Lyrebird*. No entanto, o *Grevit* possui parâmetros mais diversificados no GH, que podem ser definidos de acordo com as intenções projetuais, além de ser gratuito e com código aberto.

Por outro lado, a partir do crescente uso do ***Dynamo***, foram incorporados novos fluxos de trabalho que permitem aos usuários a troca de geometria e dados entre o *Rhino* e o *Revit* de modo mais eficiente do que a partir de *plugins* como *Grevit*, *Hummingbird* e *Lyrebird*, por exemplo. Com o ***Rhynamo***, os arquivos *Rhino* podem ser conectados às suas definições do *Dynamo* para desenvolver geometrias e controlar elementos do *Revit*. A ferramenta possui código aberto para leitura e gravação de

arquivos *Rhino* *.3dm e permite que os designers criem fluxos de trabalho bidirecionais entre os *software*. Como principais característica positivas, o *plugin* é gratuito, de código aberto e capaz de uma rápida transposição de objetos para a plataforma do *Revit*. A complexidade de uso, além da necessidade de conhecimento das sintaxes do *Dynamo* e do GH, são os fatores mais limitantes.

O ***Mantis Shrimp*** (MS), à semelhança do *Rhynamo*, também possui código aberto e conecta o *Dynamo* (*Revit*) ao *GH* (*Rhino*). Ele permite a leitura direta de arquivos nativos do *Rhinoceros* (*.3dm), bem como a geometria de exportação do GH. No entanto, diferentemente dos demais *plugins* citados, que possuem uma interface visual, o MS é escrito em *Python* na forma de objetos de usuário (do GH para exportação) e em nós *Python* personalizados (do *Dynamo* para importação), o que exige maior conhecimento de programação.

A desvantagem de todas as soluções mencionadas é que elas só funcionam com ferramentas BIM específicas. A vantagem é a compatibilidade com o sistema BIM ao qual esses *plugins* se vinculam através de suas *APIs*, permitindo maior controle ao usuário. No entanto, o compartilhamento de modelos com outros profissionais será uma colaboração baseada em arquivo e não em nenhum padrão aberto.

5.4.1.2. *Loosely Coupled*

Na “abordagem fracamente acoplada”, os sistemas são acoplados através da troca de modelos. O sistema baseado em grafos normalmente gera dados em um formato de arquivo padrão que pode ser importado diretamente no sistema BIM (Janssen *et al.*, 2016).

Essa abordagem também possui *plugins* para o GH (*Geometry Gym* e *VisualARQ*) e para o *Rhinoceros* (*VisualARQ*). Todos utilizam o IFC como formato de troca, o que possibilita que os usuários criem seus próprios procedimentos personalizados de materialização.

O ***Geometry Gym*** é um *plugin* particularmente útil para estruturas 3D e análises estruturais. Ao instalá-lo, uma guia “GG” aparece no GH com diversos componentes do *plugin*. No *Revit*, é adicionada uma guia “*GeoGym*”. Assim como os demais *plugins*, em seu funcionamento baseado na lógica de formação de entidades no *Revit*, por exemplo, linhas conectadas corretamente aos *inputs* e *outputs* do *script* GG podem gerar colunas no *Revit*. Como limitações, tem-se o fato de ele não ser gratuito e

não atuar simultaneamente com o *Revit*. Para atualizar qualquer informação ou geometria no arquivo no *Revit*, ele necessita ser importado novamente no formato IFC.

Em 2019, com o desenvolvimento do *plugin Rhino Inside Revit*, o *Geometry Gym* recebeu atualizações, com a inclusão de um novo componente "*ggRvt.Baketo-Revit*", que possibilita a conexão em tempo real com o *Revit*, utilizando sua API diretamente dentro do GH.

Por fim, o *plugin* ***VisualARQ*** possui ótimos recursos para documentação e também atua como um *add-on* do GH para programação visual e possibilidade de integração do BIM ao *Rhinoceros*. Alguns recursos, como as propriedades de materiais, por exemplo, são atribuídos somente no *Rhinoceros*, cujos arquivos são exportados em IFC. No geral, seguiu-se a mesma estrutura desenvolvida no *Geometry Gym* com algumas alterações de nomenclaturas e parâmetros. Recentemente, assim como o *Geometry Gym*, o *VisualARQ* também permite a complementação de recursos do *Rhino Inside Revit*, ou seja, a conectividade simultânea entre as ferramentas algorítmicas e BIM.

A vantagem de todos os *plugins* dessa abordagem é que eles funcionam independentemente do fluxo de trabalho, permitindo que os usuários vinculem ferramentas e sistemas para dar suporte a várias formas de colaboração e intercâmbio. No caso dos *plugins* que geram arquivos IFC padrão, os usuários têm a opção de se vincularem a qualquer aplicativo BIM que importe esse formato. No entanto, na prática, ainda existem muitos problemas com a implementação da IFC entre os principais s*oftware,* o que requer melhorias no padrão de implementações.

5.5. Modelo para testes de interoperabilidade

Com o intuito de comparar a exportação de um projeto gerado inteiramente no *Grasshopper* para o *Archicad* e para o *Revit*, foi utilizado um modelo de habitação de interesse social (GAZEL *et al.*, 2018). O modelo, denominado DOIS BITS, é parte de um trabalho sobre a reconstrução das habitações destruídas pelo desastre ambiental ocorrido no Brasil em Bento Rodrigues – MG, em novembro de 2015, após o rompimento de barragens de rejeitos de minérios na região, que deixou 226 famílias desabrigadas.

Soma-se ao desastre ambiental, ainda, um cenário de déficit habitacional no país. O Brasil é alvo há décadas de diversas políticas públicas destinadas à produção habitacional (CUNHA, 2016), mas ainda tem-se como resultado um déficit quantitativo e qualitativo de unidades.

> Em todo o país, novos bairros surgem em áreas distantes e sem urbanização, alinhando centenas de casas idênticas e minúsculas, ou enfileirando torres habitacionais com sofrível padrão construtivo, e grande impacto sobre o meio ambiente. Em face disto, a pergunta que nos vem naturalmente é: quais os resultados que essa produção provocará no cenário urbano brasileiro nos próximos anos? (FERREIRA, 2012, p.7)

Portanto, a inserção de tecnologias digitais nos processos de projeto possibilita a aplicação do conceito de customização em massa às novas habitações. Segundo Correia, Duarte e Leitão (2012), a personalização em massa permite modelos de alta qualidade a custos acessíveis, através de projeto e fabricação auxiliados por computador, o que reduz a repetição exaustiva dos processos. Ainda segundo os autores, essa abordagem supera um problema comum enfrentado pelos arquitetos ao lidar com grandes empreendimentos, e a dificuldade de projetar a variabilidade das unidades, com custo de construção sem os benefícios de economias de escala.

Além disso, a integração de processos de modelagem algorítmica e o BIM pode ser um caminho para ampliar a participação popular na tomada de decisões. Devido à facilidade de modificações nos *scripts*, é possível que os próprios moradores visualizem suas casas e definam suas prioridades habitacionais, organizadas sob a forma de parâmetros e regras aplicadas aos *scripts*.

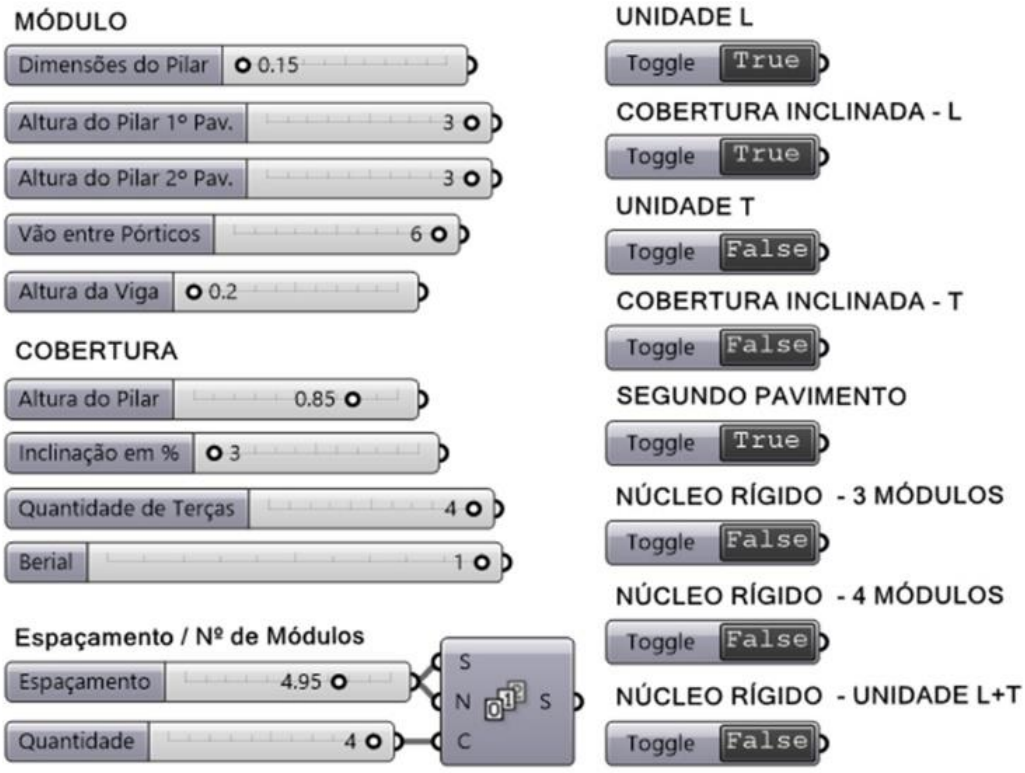
Após o levantamento do perfil populacional e padrão construtivo das unidades unifamiliares do município de Bento Rodrigues, concebeu-se um projeto de habitação com sistema estrutural em aço, com o objetivo de facilitar o transporte e a rápida construção após o processo de fabricação. Replicado de acordo com regras predefinidas, os módulos dão origem à diversas unidades distintas, mas ao mesmo tempo compatíveis entre si - a unidade mínima resulta do agrupamento de 2 módulos, e a máxima de 5. Após estudos de organização espacial, adotou-se um módulo estrutural com dimensões de 4,80m x 6,00m (comprimento x largura). As superfícies de fechamento, por sua vez, são em sistema *steel frame* (placas OSB), cujas medidas mais comuns são 1,20m x 2,40m e 1,20m x 3,00m (b x h).

Dentre as possibilidades do *script*, é possível modificar as dimensões de pilares e vigas, alturas dos pavimentos, medidas e quantidades de terças, inclinação, altura e posicionamento da cobertura, medidas dos módulos, suas quantidades e espaçamentos, disposição em formatos L, T, com segundo pavimento ou não, dentre outras possibilidades. Por outro lado, algumas restrições também foram impostas para seu correto funcionamento. As figuras 23 e 24 ilustram, respectivamente, os principais parâmetros modificáveis do *script* e algumas soluções formais encontradas a partir dele.

Quanto à construtibilidade do modelo, diversas possibilidades podem ser empregadas no *Archicad* ou no *Revit* após a exportação da unidade, no que se refere ao sistema estrutural (vigas e pilares), lajes, fechamentos externos e divisórias internas. A figura 25 ilustra o modelo testado de habitação com possibilidade mínima (2 módulos) após a renderização, já com os materiais aplicados.

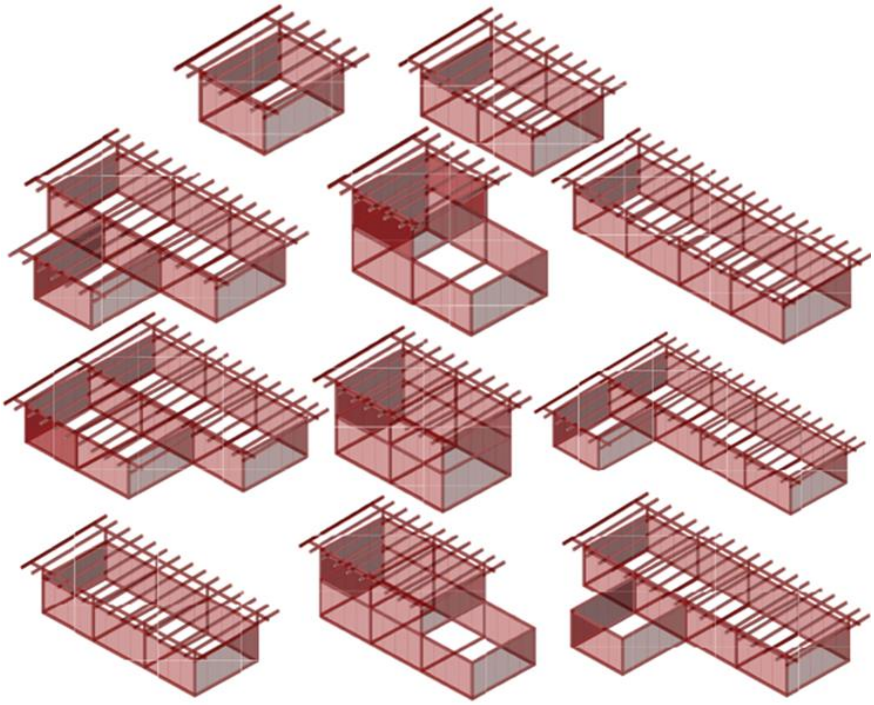
A compartimentação interna é estabelecida pela posição em planta do núcleo rígido (banheiros, lavanderia e cozinha) e, portanto, o interior é concebido como um espaço flexível, que pode ser totalmente personalizável e adaptável ao longo dos anos.

Figura 23 - Principais parâmetros modificáveis do *script* utilizado.



Fonte: Autores (2020).

Figura 24 – Diferentes tipologias geradas a partir do código visual

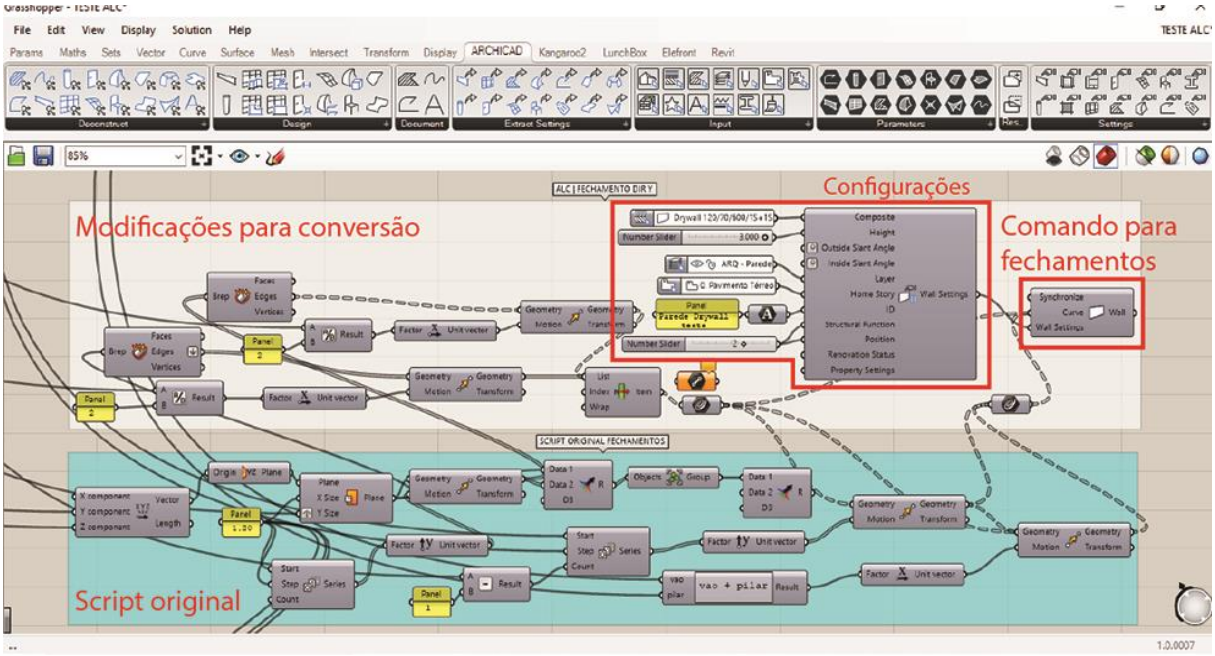


Fonte: Autores (2020).

Figura 25 – Modelo para testes renderizado, com aplicação de materiais e organização interna

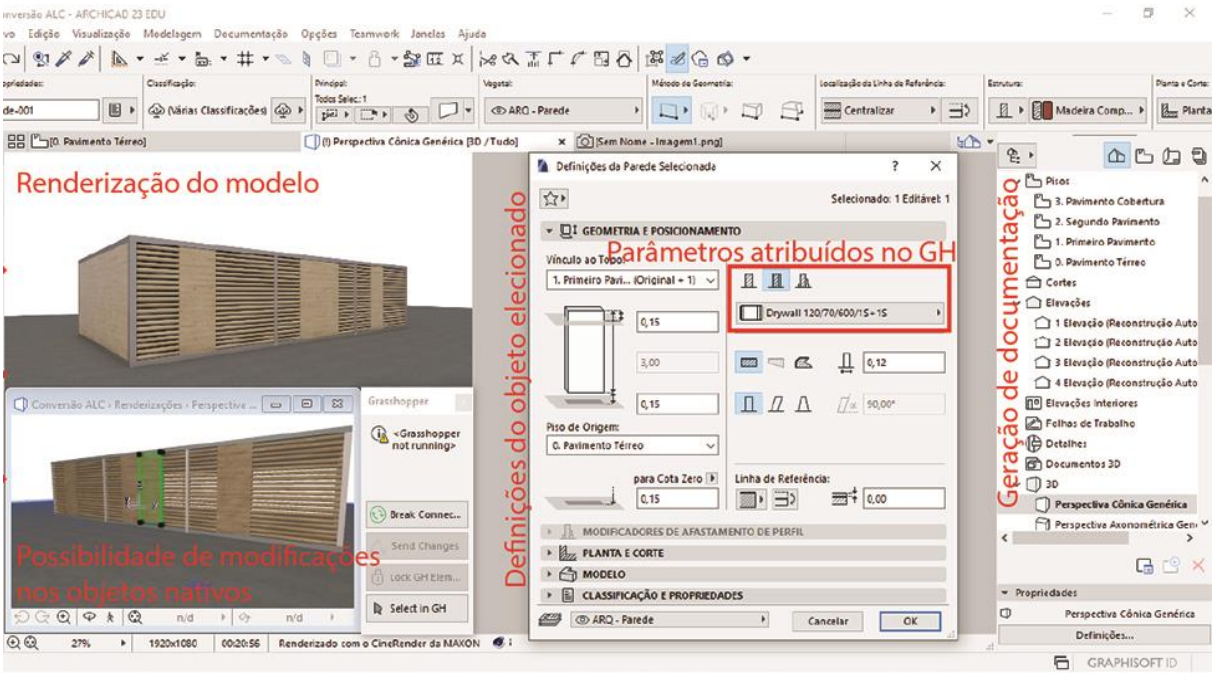
Fonte: Gazel *et al.* (2017).

Figura 26 – Interface do *Grasshoper* durante o uso do plugin *Archicad Live Connection*



Fonte: Autores (2020).

Figura 27 – Interface do *Archicad* durante o uso do *plugin Archicad Live Connection*



Fonte: Autores (2020).

5.6. Processo de exportação

Como visto, existem dezenas de *plugins* que se adequam em maior ou menor grau a diferentes necessidades projetuais. Após diversos testes, optou-se pela utilização de um *plugin* para importação no *Archicad* e outro para o *Revit*: *Archicad Live Connection* e *Rhino Inside Revit*, respectivamente. O principal diferencial de ambos é o fato de serem os únicos, dentre os testados, com conexão em tempo real com os *software* BIM.

As figuras 26 e 27 ilustram o processo de conversão do modelo utilizado. Observa-se na Figura 26 a interface do GH com o *script* original, as modificações feitas para a conversão do modelo habitacional e algumas possibilidades de configuração dos fechamentos. Na Figura 27, tem-se a interface do *Archicad* com o modelo lido como objeto nativo BIM, possibilitando modificações em suas configurações no *Archicad*, geração de documentação e geração do modelo renderizado.

Com o objetivo de exemplificar o processo de conversão do *software* de modelagem algorítmica para o BIM, será descrita resumidamente, a partir do *script* original, a lógica de formação das principais entidades utilizadas: pilares, vigas, fechamentos e lajes.

5.6.1. Pilares

Os pilares do *script* original foram feitos a partir da extrusão de um "*plane surface*" no eixo z, do piso ao segundo pavimento. Isso se repetiu movendo-se esse resultado a partir de vetores que posicionaram os pilares em suas posições corretas.

No ALC os pilares precisam ser estruturados a partir de dois pontos demarcando suas extremidades. Para tanto, a lógica foi simplesmente encontrar o centroide do plano inicial e de todas as movimentações de posicionamento dos demais pilares, bem como os pontos finais, ligando-os a uma pilha "*point*" para que sejam lidos pelo comando "*column*", do ALC. O componente "*column*" possui diversas possibilidades de configurações para os pilares, com relação a materialidade, dimensionamento, posicionamento, função estrutural, dentre outras propriedades.

No RIR os pilares são convertidos através do componente "*add column*", que recebe como *input* uma "*line*", que também pode ser obtida a partir de dois pontos.

Portanto, assim como no ALC, a lógica aplicada foi obter o centroide do “plane surface” no piso e”, movê-lo à altura do pilar no primeiro pavimento, utilizando os dois pontos para gerar uma “*line*” e ligá-la ao “*add column*”. A partir da geração do primeiro pilar, pode-se replicá-lo para as demais posições. O componente “*add column*” do RIR possui *inputs* que possibilitam selecionar o tipo de elemento (tipo de pilar usado, escolhido no Revit, e posteriormente ligado a uma categoria de modelo) e um seletor de níveis para os documentos.

5.6.2. Vigas

No script original, as vigas foram geradas de forma análoga aos pilares: através de extrusões de planos, porém nos eixos x e y. Essas extrusões foram reproduzidas com comandos “*move*” e “*series*” para obtenção de todos os trechos de vigas.

Para serem passadas ao *Archicad*, o comando “*beam*” do ALC recebe *inputs* através de curvas (ou linhas), e não através dos volumes extrudados. Novamente, as linhas foram criadas a partir dos centroides inicial e final. A mesma lógica foi aplicada às demais vigas, modeladas a partir de um comando “*line*” no componente “*beam*”. Esse componente possibilita configurações de materias para as vigas, dimensões, camadas (ou vegetais, como são chamadas no *Archicad*), ID, função estrutural, posicionamento, dentre outras propriedades.

O RIR teve a mesma lógica para a conversão dos pilares, através do componente “*add beam*”. Assim como para os pilares, o comando de vigas possui diversas possibilidades de configurações, e possibilita selecionar o tipo de elemento (tipo de viga usada, escolhida no *Revit*, e posteriormente ligada a uma categoria de modelo) e um seletor de nível de documentos.

5.6.3. Fechamentos

O *script* original gerou painéis a partir de “*plane surface*”, unidos por comandos “*merge*” e replicados através de “*move*”, ligados a expressões simples de soma de vãos com espessuras dos pilares e “*series*”.

No ALC, diferentemente da lógica utilizada no *script* original do GH, o comando “*wall*” para criação de paredes recebe como *input* as 4 arestas em vez do plano, através de “*curve*” ou “*line*”. Para essa conversão, os planos foram descontruídos em arestas para a conexão ao comando “*wall*” como curvas. Esse comando permite

modificar configurações de materialidade, dimensionamento, posicionamento, ID, função estrutural, dentre outras propriedades.

Já no RIR, o comando "*add wall*" para criação de paredes recebe como *input* apenas uma aresta da parede, através novamente de uma "*curve*" ou "*line*". Assim como no ACC, os planos foram decompostos em arestas para a conexão ao comando "*wall*" como "curve". No entanto, para simplificar o *processo*, em vez de gerar no GH vários painéis de 1,20m x 3 m cada, representando as dimensões padrões de mercado para placas OSB para estrutura em *steel frame*, optou-se por criar a linha inferior na dimensão dos módulos (de 6 m), que posteriormente poderão ser divididos em painéis com as dimensões desejadas no próprio Revit, ao especificar o material utilizado.

O componente "*add wall*" do RIR possibilita configurações do tipo de fechamento usado, escolhido no *Revit*, do nível, da altura da parede e do posicionamento da linha de eixo. Permite também a inversão de faces interna e externa, a união das extremidades e a seleção entre parede estrutural ou não de acordo com os esforços atribuídos.

5.6.4. Pisos e lajes

Como o *script* original não possuía lajes, optou-se por criá-las para complementar o projeto, utilizando-se a mesma lógica dos fechamentos, a partir de planos, séries e movimentações para posicionamento correto.

No ALC, o comando para lajes do ACC "*slab*" recebe como *input* polígonos, criados pelo comando "*geometry*". Para transformar as lajes nessas geometrias, foi utilizada a mesma lógica dos fechamentos, descontruindo-se os planos em arestas para a conexão. O componente de lajes pode ser ligado ao "*slab settings*", permitindo modificações de materias, dimensões, ângulo de referência, camadas (vegetais), ID, função estrutural, posicionamento, dentre outras propriedades.

Já o componente para lajes do RIR "*add floor*" recebe como *input* um "*boundary*", através de uma "*curve*" ou "*surface*" por exemplo. Isso possibilitou ligar diretamente os planos da laje, replicando-os e movendo-os para o correto posicionamento em ambos pavimentos. Esse componente de paredes do RIR pode ser configurado com relação ao tipo de fechamento usado, ao nível, e à função, sendo estrutural ou não.

5.7. Resultados preliminares

Após as conversões realizadas em ambos *plugins*, foram analisadas suas potencialidades e limitações. Convém destacar que algumas análises são subjetivas e dependem da familiaridade com cada *software* BIM e sua interface.

Em ambos os casos, todas as entidades geométricas foram criadas a partir de pontos, curvas ou superfícies, o que está diretamente relacionado à lógica de desenho em cada *software* BIM. No *Revit*, por exemplo, as paredes são desenhadas a partir de linhas, tornando-se também a geometria utilizada no GH para leitura do *plugin* de conversão do RIR.

Com relação às interfaces, considerou-se a interface do RIR mais completa, uma vez que apresenta maiores informações a serem atribuídas a cada elemento, além da organização em listas para escolha de dados, semelhante a uma extensão do *Revit* dentro do GH. A interface do ALC no GH tampouco é complexa, mas utiliza pouco as listas, o que pode torná-la menos intuitiva. No entanto, o ALC consome menor tempo de processamento, tornando a conexão instantânea entre *software*. O RIR, por sua vez, demanda um dispêndio computacional maior, o que exige uma máquina com capacidade de processamento mais robusta.

Para o RIR, as medidas das seções transversais do *script* original dos pilares, vigas, fechamentos e lajes são desnecessárias no modelo final exportado, uma vez que o *plugin* utiliza os atributos e métricas relacionados às famílias de entidades disponíveis no *Revit*. Todas as famílias de objetos exibidos no modelo no *Revit*, tornam-se opções listadas também no GH.

O *Archicad*, por sua vez, permite a utilização das dimensões do *script* original ou a escolha de qualquer entidade (laje, viga, pilar, etc) existente como famílias no *Archicad*.

No *Archicad* também é possível escolher se a entidade será gerada no eixo central, topo ou base da seção transversal, o que facilita a junção entre pilares e vigas, por exemplo. No *Revit*, no entanto, as entidades são sempre geradas a partir do eixo central da geometria, exceto os planos de paredes.

Caso as entidades sejam desbloqueadas no *Revit*, as alterações posteriores no *script* do GH não são computadas neste *software* BIM. No entanto, observou-se um maior controle dessas alterações no *Archicad* após o desbloqueio.

O modelo construtivo resultante no *Archicad* e no *Revit* pode ser aprimorado em ambas interfaces após suas conversões. Para facilitar a compreensão por parte dos usuários, é possível produzir imagens tridimensionais que indiquem acabamentos e detalhes construtivos. Com a documentação gerada, também pode-se obter uma estimativa da quantidade e dos custos dos materiais de construção.

A possibilidade de usar componentes diferentes depende da biblioteca de objetos que cada usuário possui, tanto para o *Archicad* quanto para o *Revit*. No *Revit*, para o carregamento de novos elementos, deve-se “recomputar”, para que eles apareçam nas listas do GH. Porém, cada vez que se recomputam os dados, o RIR cria elementos de revisão duplicados no *Revit*. A mesma limitação de sobreposição ocorre ao abrir um arquivo salvo no GH e outro no *Revit*. Isso inviabiliza processos primordiais, como por exemplo a utilização de novas famílias adicionadas ao *Revit*. Ao abrir somente o arquivo salvo do GH, essas famílias do *Revit* não são carregadas na lista do GH.

Portanto, essa limitação inviabiliza o uso eficiente do *plugin* RIR. Ao contestar os desenvolvedores sobre esse problema em um fórum de uma plataforma colaborativa, eles garantiram que estão tentando corrigir desde fevereiro de 2020, e espera-se que, em breve, essa falha seja solucionada.

A partir dos resultados expostos, embora o RIR tenha um enorme potencial, considerou-se o ALC mais eficiente até o momento para as transferências de dados entre o GH e o *Archicad,* como um *software* BIM.

5.8. Discussões e trabalhos futuros

A decisão sobre a transição entre plataformas geralmente resume-se à necessidade de transferência de dados. No entanto, de acordo com o objetivo do projeto, em determinados momentos é válido importar geometrias sem o consumo de tempo para traduzir os dados atribuídos a estas. Em outros, é importante criar geometria e dados nativos para utilizar as ferramentas de documentação de *software* BIM ou permitir a edição do modelo após sua inserção. Para projetos que exigem geometria nativa, é importante adotar o método mais eficiente para percorrer as plataformas de *software,* sem o retrabalho de modelagem da geometria e inserção de dados.

Nesse cenário, os campos de projeto e construção tem se dedicado a alternativas para os problemas de interoperabilidade e compatibilidade de *software*. Ao longo da pesquisa, observou-se que todas as soluções discutidas estão continuamente em modificação para atualização e correção de erros em *plugins*, seja a partir de discussões em plataformas colaborativas, em iniciativas *open source* ou de *software* proprietários.

Por sua vez, o recurso de portabilidade dos *plugins* do *Grasshopper* possui algumas limitações. Em diversos casos, partes significativas do *script* não são portáteis e várias operações são refeitas para a geração da geometria em diferentes *software*. Neste sentido, apesar dos diversos *plugins* analisados possuírem diferentes interfaces e protocolos para transferência dos dados, nenhum é perfeitamente eficiente em todos os aspectos. As barreiras de interoperabilidade devem-se em parte às limitações da API do *software* BIM e também à falta de desenvolvimento do formato de arquivo IFC, que busca interoperabilidade universal no setor de AEC.

Adicionalmente, há barreiras culturais para a utilização integrada de diversos *software* em um mesmo modelo de projeto. Como consequência, há reconstrução das geometrias em *software* BIM, que além do retrabalho, resultam na perda de grande parte da inteligência incorporada ao *design* original (Kaushik, 2017). Neste caso, o resultado final pode retroceder às limitações de modelagem do *software* BIM, em um processo que contradiz os princípios conceituais da modelagem da informação da construção.

Nesse contexto, como trabalhos futuros, objetiva-se aprofundar os estudos sobre a interoperabilidade do projeto algorítmico ao BIM e o seu potencial de aplicação em projetos de customização em massa na arquitetura. Pretende-se analisar também a utilização de nuvens no processo projetual, possibilitando o trabalho compartilhado e integrado através de computadores em diferentes localidades.

Por fim, tendo em vista o projeto de unidades de habitação de interesse social, vislumbra-se que, com o domínio e integração das plataformas, os arquitetos possam criar interfaces personalizadas e intuitivas de *design* arquitetônico interativo. Essas ferramentas permitirão aos usuários aplicar regras para personalizar seus interesses projetuais, dentro de possibilidades codificadas por um modelo predefinido, bem como especificar dimensões e acabamentos que possam atender às suas restrições econômicas.

Agradecimentos

Os pesquisadores agradecem a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro à pesquisa, ao Laboratório de Modelagem Digital Nó.Lab e ao Programa de Pós-Graduação em Arquitetura e Urbanismo da UFV.

Referências

BIANCONI, F., FILIPPUCCI, M., BUFFI, A. (2019). **Automated design and modeling for mass-customized housing.** A web-based design space catalog for timber structures. *Automation in Construction*, *103*(March), 13–25. https://doi.org/10.1016/j.autcon.2019.03.002

CASTELO BRANCO, R., LEITÃO, A. (2017). **Integrated algorithmic design:** A single-script approach for multiple design tasks. *ECAADe 35 - Design Tools - Theory*, *1*, 729–738. https://doi.org/10.1063/1.1649724

CORREIA, R., DUARTE, J., LEITÃO, A. (2012). **GRAMATICA:** A general 3D shape grammar interpreter targeting the mass customization of housing. *Digital Physicality - Proceedings of the 30th ECAADe Conference - Volume 1 / ISBN 978-9-4912070-2-0, Czech Technical University in Prague, Faculty of Architecture (Czech Republic) 12-14 September 2012*, *1*(Knight), 489–496.

CUNHA, T. F. (2016). ***Editais , contratos e medições do Programa Minha Casa Minha Vida (PMCMV) com base nos critérios de desempenho da norma NBR 15 . 575 / 2013 da Associação Brasileira de Normas Técnicas (ABNT)***.

FEIST, S. T. de V. (2016). ***A-BIM :*** *Algorithmic-based Building Information Modelling*. *May*.

FERREIRA, J. S. W. (Coord. . (2012). **Produzir casas ou construir cidades?** Desafios para um novo Brasil urbano. Parâmetros de qualidade para a implementação de projetos habitacionais e urbanos. In *Surveillance and Society* (Vol. 2, Issues 2–3).

GAZEL, J. L., MARTINEZ, A. C. P., DOS SANTOS, D. M., LOPES DE SOUZA, D. (2018). ***2 BITS:*** *A case of mass customization for social housing*. 353–358. https://doi.org/10.5151/sigradi2018-1744

GUIDOUX GONZAGA, M., VELOSO, L. P., PAIVA PONZIO, A., MIOTTO BRUSCATO, U. (2018). ***Cutting the Path:*** *Encouraging Formal Exploration Through Integration Between Algorithmic and BIM Environments*. 11–16. https://doi.org/10.5151/sigradi2018-1275

JANSSEN, P., CHEN, K. W., MOHANTY, A. (2016). **Automated Generation of BIM Models.** *ECAADe*, *2*, 583–590. http://papers.cumincad.org/cgi-bin/works/Show?ecaade2016_239

KAUSHIK, V. (2017). ***Why do Architects need Computational BIM Workflows?*** https://wowad.in/why-do-we-need-computational-bim-workflows/?utm_campaign=Thank God It%27s Computational&utm_medium=email&utm_source=Revue newsletter

LARRONDO, A. (2017). **Generación y control de formas libres en entornos BIM :** modelado paramétrico, modelado algorítmico. *TDX (Tesis Doctorals En Xarxa)*. http://www.tdx.cat/handle/10803/457875

SCHUMACHER, P. (2016). ***Design Parameters to Parametric Design*****.** http://www.patrikschumacher.com/Texts/Design Parameters to Parametric Design.html

6. CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo principal testar diferentes alternativas para a interoperabilidade durante o processo de projeto através da utilização de diferentes *software* em múltiplos fluxos de trabalho.

Nesse contexto, a fundamentação teórica se pautou na história sobre a difusão de diferentes métodos de projeto, o chamado *Design* Computacional. Observou-se que, embora os recursos de exploração, geração, reflexão e modificação de processos de projeto tradicional em papel continuem sendo utilizados nos processos computacionais, a lógica e os componentes sequenciais do processo de projeto foram transformados.

O *design* computacional pode auxiliar o arquiteto em automatizar tarefas, estender funcionalidades dos *software* de modelagem, fazer alterações e negociar a interrelação de conjuntos de dados de informação que são relevantes para o projeto arquitetônico, melhorando assim sua capacidade de projetar.

Nesse cenário, foram analisados o *design* paramétrico e o *design* algorítmico. A partir das definições de diversos autores, o conceito de *design* paramétrico no qual se pautou esse trabalho, parte simplesmente da descrição de um projeto simbolicamente baseado no uso de parâmetros. Portanto, constatou-se que qualquer projeto de arquitetura que se utilize de parâmetros em sua concepção, é um projeto paramétrico, desde os tratados de Alberti, Palladio e Durand.

No entanto, no contexto aqui discutido, das tecnologias computacionais, considerou-se a definição de projeto paramétrico auxiliado por computador, cuja única premissa é a utilização de alguma ferramenta computacional auxiliando em um processo de projeto baseado na atribuição de parâmetros. Esses *software* paramétricos, trazem como principal funcionalidade a possibilidade de atribuição de valores aos parâmetros e sua manipulação, estabelecendo associações entre eles.

Foi constatado que essas associações se traduzem, por um lado, nas relações paramétricas encontradas em modelagens de *software* BIM, como o *Archicad* e o *Revit*, e, por outro, nas modelagens algorítmico-paramétricas de ferramentas de linguagem de programação visual, como o *Grasshopper*.

Com relação aos conceitos de *design* algorítmico discutidos, chegou-se à sua definição como um paradigma de *design* que utiliza algoritmos para gerar modelos. A partir das definições de Leitão e Santos (2011), considerou-se que todo *design*

algorítmico é também um *design* generativo. Porém, no design algorítmico existe uma correlação entre o algoritmo e o modelo gerado, que proporciona rastreabilidade e permite ao usuário identificar as partes do algoritmo que geraram determinada parte do modelo.

A associação do *design* paramétrico ao algorítmico também foi abordada, sendo exemplificada pelo processo de projeto que ocorre no *Grasshopper*. Nele, o *design* paramétrico se baseia no processo de exploração e reedição de relacionamentos associativos em um espaço de solução geométrica. Nesse processo, o arquiteto projeta um código para modelar a geometria desejada. Esse esquema paramétrico suporta processos algorítmicos de geração de formas.

Nesse contexto, a arquitetura passou a ser resultado da integração do arquiteto com ferramentas computacionais que auxiliam nos processos envolvidos na concepção, desenvolvimento e ciclo de vida do projeto. Observou-se a necessidade entre os profissionais de AEC, do conhecimento sobre os novos métodos, técnicas e ferramentas com os quais eles passam a interagir através do *design* computacional.

É comum que o projeto se adapte as ferramentas disponíveis dentro de *software* de desenho CAD e BIM, uma vez que as funcionalidades disponíveis nesses programas são limitadas. Como forma de superar essas barreiras, o uso da programação no processo de projeto pode auxiliar na adaptação de *software* à metodologia estabelecida pela equipe e às diretrizes de projeto.

De modo geral, o arquiteto com capacidade de programar seus próprios recursos pode desenvolver habilidades compatíveis com as suas novas responsabilidades na era computacional, o que o coloca ao centro dos processos de informação. Assim, ele pode não somente automatizar tarefas, mas também integrar informações para aprimorar ou até mesmo criar novos processos durante o projeto, utilizando efetivamente o potencial de controle do BIM para integrar, compartilhar e monitorar o atendimento de regras em uma edificação.

No entanto, o campo de conhecimento de programação, muitas vezes não faz parte da formação profissional dos arquitetos. A discussão da relação entre as áreas de arquitetura e computação sugere que uma progressão nas habilidades de programação de arquitetos é desejável, para que possam formular e modelar problemas que pretendem abordar de maneira mais eficiente e com maior autonomia.

Como alternativa para facilitar o uso da programação, tem-se a programação visual, mais intuitiva e dinâmica para os processos de projeto. Essa tem se tornado

uma competência de grande importância para o futuro da profissão, quando se considera que os processos de projeto contam com cada vez mais requisitos e particularidades que dificilmente são atendidos por grandes desenvolvedores.

Dentro do estudo das linguagens de programação visual, foram estudadas as relações dos processos de projeto paramétricos e algorítmicos com diversos *software* e *plugins* associados a esses tipos de *Design*, destacado os *software* BIM *Archicad* e *Revit* e a ferramenta de modelagem algorítmico-paramétrica *Grasshopper*. A compreensão das principais potencialidades e limitações de cada *software* permitiu a proposição de fluxos de trabalho baseados na complementação entre eles. Isso pode ser resumido pela integração da liberdade formal propiciada pelo *Grasshopper*, sobretudo nas etapas de concepção projetual, associada à metodologia organizacional BIM, que, quando bem implementada, propicia uma eficiente gestão do projeto.

Antes de aprofundar na forma como os *software* se complementam de forma prática, foi importante entender as relações deles com os processos de projeto mencionados, visando compreendê-los sob o ponto de vista da teoria da arquitetura, na qual o "*design*", podendo ser traduzido como o processo de projeto, é estudado como um tema de pesquisa.

Constatou-se que, embora qualquer *software* seja formado por algoritmos, que processam dados de entrada e obtêm dados de saída, o *Archicad* e o *Revit* não são considerados *software* algorítmicos, são paramétricos. No entanto, o *Archicad* traz como possibilidade para satisfazer às necessidades de criação e desenvolvimento de formas complexas, a conexão com o *Rhinoceros* e com o *Grasshopper,* através do *plugin Archicad Live Connection*.

Já o *Revit*, possui uma interface gráfica algorítmica, baseada em linguagem de programação visual, o *Dynamo*. Porém, essa interface possui limitações quando comparada ao *Grasshopper*, tanto na modelagem de geometrias, quanto na interoperabilidade com diversos *plugins* desenvolvidos nos últimos anos.

Embora os dois primeiros (*Archicad* e *Revit*) sejam *software* BIM e o terceiro (*Grasshopper*) seja uma extensão do CAD (através do *Rhinoceros*), o *Archicad*, o *Revit* e o *Grasshopper* possuem em comum a atribuição de parâmetros durante todo o processo de projeto, o que configura essas três ferramentas como paramétricas. No entanto, essa atribuição não segue uma mesma lógica em *software* BIM e de modelagem algorítmico-paramétrica baseada em VPL.

O *Grasshopper* permite que os processos de atribuição de parâmetros sejam visualizados e manipulados, para que possam ser obtidas mais variações durante a modelagem da geometria, com maior facilidade e com uma gama superior de possibilidades formais do que em *software* BIM. Portanto, esse sistema é especializado na descrição dos processos de projeto para a geração dos resultados. Isso possibilita reescrever o histórico de uma performance a qualquer momento, alterando partes do modelo e observando suas consequências. Conforme as classificações de Janssen, as modelagens desenvolvidas nessas ferramentas são baseadas em fluxos de dados.

Por outro lado, os sistemas BIM são classificados como de modelagem associativa, por permitirem que os objetos modelados possam ser modificados através da mudança de seus parâmetros, sem a necessidade de remodelá-los, uma vez que as famílias e geometrias possuem parametria. Com relação à modelagem, esses sistemas são mais orientados para oferecer uma boa interação do resultado final em detrimento desse controle das relações internas que o originaram.

No entanto, ao se analisar a metodologia BIM a partir de suas outras perspectivas (além da modelagem), aplicadas no *Archicad* ou no *Revit*, um dos pontos de destaque é justamente o amplo controle de todo o processo de projeto. Esse controle pode ser exemplificado através de diversos recursos, entre eles: utilização de um modelo central que pode ser compartilhado, e reúne todas as informações do projeto; possibilidade de criação de vínculos para conciliar os projetos feitos em modelos separados; acesso ao mesmo modelo por parte de toda a equipe, com acréscimo de informações ao longo de todo o processo de projeto; alterações realizadas no modelo são transmitidas automaticamente para todos os documentos, tabelas e planilhas gerados; verificação de interferências (*clash detection*) no modelo.

Portanto, a execução em conjunto da interface do usuário de um *software* de modelagem algorítmica (*Grasshopper*) e um *software paramétrico* BIM (*Archicad* ou *Revit*) se mostrou eficaz na concepção formal do modelo durante o processo de projeto. O *Grasshopper* fornece um ambiente de programação visual para definir a lógica algorítmica e conceber um modelo. Elementos BIM nativos podem ser manipulados a partir do *Grasshopper*, com seus metadados incluídos E muitas das formas complexas que são difíceis de gerar no *Archicad* ou no *Revit* podem ser desenvolvidas utilizando o *Grasshopper*. Por outro lado, os *software* BIM são muito mais especializados na gestão do projeto, documentação e adição de informações por toda a equipe, trabalhando colaborativamente. Os levantamentos de quantidade e cronogramas tornam-

se parte do processo de projeto desde o início, auxiliando em todas as etapas. As ferramentas de análise do *Grasshopper* também podem se tornar parte do processo de projeto.

A partir da compreensão dos conceitos de *design* computacional e dos *software* mencionados, essa pesquisa buscou investigar maneiras para a integração de diversos *software*, específicos para cada necessidade durante os processos projetuais, possibilitando maior controle do processo de projeto desde o desenvolvimento das modelagens, até a organização das documentações.

Os testes realizados mostraram que, em determinados momentos é válido importar geometrias sem o consumo de tempo para traduzir os dados atribuídos a estas. Mas, em outros, é importante criar geometria e dados nativos para utilizar as ferramentas de documentação de *software* BIM ou permitir a edição do modelo após sua inserção. Para projetos que exigem geometria nativa, é importante adotar o método mais eficiente para percorrer as plataformas de *software*, sem o retrabalho de modelagem da geometria e inserção de dados. Nesse contexto, foram apresentadas as principais característica e pontos positivos e negativos de diversos *plugins* ativos de interoperabilidade entre *software* de modelagem algorítmico-paramétrica e BIM.

Ao longo da pesquisa, observou-se que campos de AEC vem se dedicando a alternativas para esses problemas de interoperabilidade e compatibilidade de *software*, a partir de discussões em plataformas colaborativas, em iniciativas *open source* ou de *software* proprietários. Foram criados nos últimos dois anos diversos *plugins*, como o próprio *Rhino Inside Revit*, utilizado nessa pesquisa, e corrigidos inúmeros erros em vários *software* e *plugins*, além de efetuadas diversas atualizações e aprimoramentos nos programas existentes durante o período dessa pesquisa.

No entanto, observou-se que o interesse das empresas BIM proprietárias dos *software* ainda é bastante restrito a elas mesmas, não focando na intercomunicação com outras empresas ou áreas externas. Tomando como exemplo a Autodesk, ela possui plataformas em diversos setores de AEC e, com isso, consegue acompanhar os projetos durante todo seu ciclo de vida tanto na arquitetura, quanto na engenharia e em outras operações. No entanto, um escritório que se utilize de *software* da *Autodesk*, em determinados momentos pode apresentar necessidade de se comunicar com outro que se utilize da *Graphisoft*, por exemplo. Nesse cenário, ainda faltam mecanismos para que haja uma perfeita interoperabilidade dos projetos desenvolvidos a partir de ambas as empresas proprietárias.

O recurso de portabilidade dos *plugins* do *Grasshopper* também demonstrou algumas limitações. Em diversos casos, na tentativa de interoperabilidade com os *software* BIM, partes significativas do código visual não foram enviadas e várias operações tiveram que ser refeitas para a geração da geometria em diferentes *software*. Neste sentido, apesar dos diversos *plugins* analisados possuírem diferentes interfaces e protocolos para transferência dos dados, nenhum foi considerado perfeitamente eficiente em todos os aspectos.

Com relação às barreiras de interoperabilidade via IFC, é certo que falta de interesse das empresas desenvolvedoras de software BIM em implementar bons tradutores IFC para a transmissão de dados a *software* de outras empresas.

Como visto, ao longo desse trabalho foram realizados diversos testes a partir de uma variedade de ferramentas (aplicativos, *software*, *plugins*, etc.), que visam solucionar algumas lacunas relacionadas ao processo de projeto em BIM. Com isso, foram levantadas inúmeras possibilidades para ampliar as potencialidades desses processos. No entanto, todas conduziram à conclusão de que os escritórios de arquitetura ainda se veem restritos à escolha de uma empresa proprietária para seus projetos e, quando necessário o intercâmbio com outro escritório, há dificuldades de comunicação e de transmissão de dados.

Os testes desenvolvidos investigaram as possibilidades de integração dentro desse cenário, não deixando que as limitações impostas por algumas barreiras que naturalmente surgiram com os novos processos de projeto a partir do uso do BIM limitassem o uso de *software* específicos para cada necessidade de projeto.

REFERÊNCIAS

AISH, Robert. **First Build Your Tools.** Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design, p. 36–49, 2013a.

AISH, R Robert. **DesignScript:** a learning environment for Design Computation. In: Design Modelling Symposium, Berlim, 2013b.

AISH, Robert. **DesignScript:** Scalable Tools for Design Computation. Computation and Performance - Proceedings of the 31st eCAADe Conference, v. 2, p. 87–95, 2013c.

AISH, R.; HANNA, S. **Comparative evaluation of parametric design systems for teaching design computation.** Design Studies, v. 52, p. 144–172, 2017. Disponível em: http://dx.doi.org/10.1016/j.destud.2017.05.002. Acesso em: jan. 2022.

ALEXANDER, C. **Notes on the Synthesis of Form.** Harvard University Press, 1964.

ANDRADE, Max Lira. **O Trabalho de Localização do ArchiCAD BRA, por Max Andrade.** GraphisoftBR, 2014. Disponível em: https://graphisoftbr.blogspot.com/2014/08/conheca-o-trabalho-de-localizacao-do.html. Acesso em: out. 2019.

ARAUJO, André L.; CELANI, Gabriela. **Exploring Weaire-Phelan through Cellular Automata:** A proposal for a structural variance-producing engine. Blucher Design Proceedings, v. 3, n.1, p. 710-714, 2016.

ARCHER, B. **Systematic method for designers.** The Design Council, Londres, 1965. Reimpresso em N. Cross (ed) (1984). Developments in design methodology. John Wiley, Chichester.

BIM FÓRUM BRASIL, **Histórico BIM**, 2019. Página inicial. Disponível em: https://www.bimforum.org.br/historico. Acesso em: jun. 2019.

BRASIL. **Decreto nº 10.306, de 2 de abril de 2020.** Dispõe sobre a Estratégia Nacional de Disseminação do Building Information Modeling. Disponível em: https://www.in.gov.br/en/web/dou/-/decreto-n-10.306-de-2-de-abril-de-2020-251068946. Acesso em: set. 2021.

BRAVO MARTÍNEZ, Maite. **Lógicas paramétricas en la arquitectura del siglo XX.** Tese doutoral, UPC, Departament de Projectes Arquitectònics, Catalunia2016. Disponível em: http://hdl.handle.net/2117/96225 Acesso em: out. 2021.

BRÍGITTE, Giovanna. **Parâmetros de projeto, BIM e aprendizado de máquina no suporte à decisão projetual.** 2019. Disponível em: http://ejournal.unp.ac.id/index.php/tingkap/article/view/1881/1614. Acesso em: mai. 2021.

BURNETT, M.; BAKER, M.; BOHUS, C.; CARLSON, P.; YANG, S.; VAN ZEE, P. **Scaling up visual programming languages.** Computer, v. 28, n. 3, p. 45–54, 1995. Disponível em: http://ieeexplore.ieee.org/document/366157/. Acesso em: set. 2021.

BURRY, M. **Narrowing the Gap between CAAD and Computer Programming.** Proceedings of the CAADRIA Conference, 1997.

BURRY, M. **Dimensions. In: Scripting Cultures:** Architectural Design and Programming. Architectural Design Primer, p. 86-125, 2011.

CAETANO, Inês; SANTOS, Luís; LEITÃO, António. **Computational design in architecture:** Defining parametric, generative, and algorithmic design. Frontiers of Architectural Research, v. 9, n. 2, p. 287–300, 2020.

CASTELO BRANCO, Renata; LEITÃO, António. **Integrated algorithmic design:** A single-script approach for multiple design tasks. eCAADe 35 - Design Tools - Theory, v. 1, p. 729–738, 2017. Disponível em: http://papers.cumincad.org/data/works/att/ecaade2017_031.pdf. Acesso em: ago. 2021.

CASTELLS, Manuel. **La Era de la Información: Economía, Sociedad y Cultura.** Vol. 1. México : Siglo XXI, 1996.

CELANI, M. G. C.; VAZ, C. E. V. CAD **Scripting And Visual Programming Languages For Implementing Computational Design Concepts:** A Comparison From A Pedagogical Point Of View. International Journal of Architectural Computing, v. 10, n. 01, p. 121–138, 2012

CELANI, G.; SEDREZ, M.; LENZ, D.; MACEDO, A. **The Future of the Architects's Employment:** To Which Extent Can Architectural Design be Computerised? In: G. Celani; D. M. Sperling; J. M. S. Franco (Orgs.), Computer-Aided Architectural Design. The Next City - New Technol-ogies and the Future of the Built En-vironment. 16th International Conference, CAAD Futures 2015, São Paulo, Brasil, p.195-212, 2015.

CÔCO JÚNIOR, Verley Henry; CELANI, Gabriela. **From the automated generation of layouts to fabrication with the use of BIM:** a new agenda for Architecture in the 21st century. p. 23–30, 2018.

DAVIDSON, S. **Grasshopper - Algorithmic Modeling,** 2015. Disponivel em: http://www.grasshopper3d.com/page/architectureprojects>. Acesso em: Out. 2020.

EASTMAN, Chuck. **Spatial Synthesis in Computer-Aided Building Design.** Applied Science, London, 1975.

EASTMAN, Chuck. **Report on Integrated Practice. American Institute of Architects**, 2006.

EASTMAN, Chuck. *et al.* **Manual de BIM:** um guia de modelagem da informação da construção para arquitetos, engenheiros, gerentes, construtores e incorporadores. Porto Alegre: Bookman, 2014.

ENGELBART, Douglas C. **Augmenting human intellect:** a conceptual framework. Stanford Research Institute, Califórnia, 1962 . Disponível em: https://dougengelbart.org/content/view/138#9 Acesso em: out. 2021.

FERREIRA, B.; LEITÃO, A. **Generative Design for Building Information Modeling.** Proceedings of the 33rd eCAADe Conference, Vienna, Austria, v. 1, p.635–644, 2015.

GUIDOUX GONZAGA, Mário *et al.* **Cutting the Path:** Encouraging Formal Exploration Through Integration Between Algorithmic and BIM Environments. p. 11–16, 2018.

HEVNER, A.R.; MARCH, S.T.; PARK, J.; RAM, S. **Design science in Information Systems Research.** MIS Quarterly, v. 28, n. 1, p. 75-105, 2004.

HOLZER, D. **BIM and Parametric Design in Academia and Practice:** The Changing Context of Knowledge Acquisition and Application in the Digital Age. International Journal of Architectural Computing, v. 13, n. 1, p. 65–82, 2015.

HUAHUI, Lai; DENG, Xueyuan. **Interoperability analysis of IFC-based data exchange between heterogeneous BIM software.** Journal of Civil Engineering and Management. 2018 Volume 24 Issue 7: 537–555, 2018. Disponível em: https://doi.org/10.3846/jcem.2018.6132 Acesso em: jan. 2022.

JANSSEN, P.; STOUFFS, R. **Types of parametric modelling.** CAADRIA 2015 –20th International Conference on Computer-Aided Architectural Design Research in Asia: Emerging Experiencesin the Past, Present and Future of Digital Architecture, n. May, p. 157–166, 2015

JANSSEN, P., CHEN, K. W., & MOHANTY, A. **Automated Generation of BIM Models**. ECAADe, 2, 583–590, 2016. Disponível em: http://papers.cumincad.org/cgi-bin/works/Show?ecaade2016_239 Acesso em: jul. 2019.

JANSSEN, P.; LI, R.; MOHANTY, A. **MÖBIUS:** A parametric modeller for the web. Living Systemsand Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference on Computer-Aided Architectural Design Research in Asia CAADRIA, 2016.

JONES, J. C. **A Method of Systematic Design**, in Conference on design methods, eds J. C. Jones and D. G. Thornley, The Macmillan Company, New York, 1963.

KALAY,Y. E. **Architecture´s New Media:** Principles, Theories, and Methods of Computer-Aided Design. London: MIT Press, 2004. 536p.

KASSEM M., KELLY G, DAWOOD N, *et al.* **BIM in facilities management applications:** a case study of a large university complex. Built Environ Proj Asset Manag 2015; 5: 261–277.

KEOUGH, I.; HAUCK, A. **From Pencils to Partners:** The Next Role of Computation in Building Design. In: R. Garber (Org.). Workfl ows: Expanding Architecture's Territory in the Design and Delivery of Buildings. p.74–81, 2017.

KIERAN, AJ Stapleton; BARRY, J. Gledson; ZAID, Alwan. **Entendendo a interoperabilidade tecnológica através de observações de vazamento de dados em transações baseadas em Modelagem de Informações da Construção (BIM)**. Fusion,

Anais da 32ª Conferência Internacional sobre Educação e pesquisa em Projeto de Arquitetura Assistida por Computador na Europa , 515-524. Vol. 2. eCAADe: Conferences 2. Newcastle upon Tyne, Reino Unido: Northumbria University, 2014. Disponível em: http://papers.cumincad.org/data/works/att/ecaade2014_052.content.pdf. Acesso em: abr. 2019.

KOLAREVIC, B. **Architecture in the Digital Age:** Design and Manufacturing. New York and London, Taylor & Francis, 2003.

LANDIM, Gabriele do Rosário; SPERLING, David Moreno. **Programação para Arquitetura:** linguagens visuais e textuais em Projeto Orientado ao Desempenho. 2019.Universidade de São Paulo, São Carlos, 2019. Disponível em: http://www.teses.usp.br/teses/disponiveis/102/102131/tde-09092019-100632/pt-br.php. Acesso em: ago. 2021.

LARRONDO, Antonio. **Generación y control de formas libres en entornos BIM.** Modelado paramétrico, modelado algorítmico. Universitat Potitècnica de Catalunya (UPC), Espanha, 2017. Disponível em: https://upcommons.upc.edu/handle/2117/112206. Acesso em: jul. 2019.

LEITÃO, A.; CABECINHAS, F.; MARTINS, S. **Revisiting the Architecture Curriculum:** The programming perspective. 28th eCAADe Conference Proceedings, p. 81–88, 2010.

LEITÃO, A.; SANTOS, L. **Programming Languages For Generative Design:** Visual or Textual?eCAADe 29, n. 01, p. 549–557, 2011.

LEITÃO, António. **Teaching Computer Science for Architecture:** a proposal. Future Traditions: 1ST eCAADe Regional International Workshop, p. 95–104, 2013.

LIU, Shijing; XIE, Benzheng; TIVENDALE, Linda; LIU, Chunlu. **The driving force of government in promoting BIM implementation**, Journal of management and sustainability, vol. 5, no. 4, Toronto, 2015, p. 157-164. Disponível em: http://dro.deakin.edu.au/view/DU:30080934. Acesso em: jul. 2019.

LOVE, P. *et al.* **BIM in the operations stage: bottlenecks and implications for owners.** Built Environment Project and Asset Management, 2015.

LUKKA, K. **The constructive research approach.** In Ojala, L. & Hilmola, O-P. (eds.) Casestudy research in logistics. Publications of the Turku School of Economics and Business Administration, Series B1, p.83-101, 2003

MANZIONE, Leonardo. **Proposição de uma estrutura conceitual de gestão de processo de projeto colaborativo com o uso do BIM.** Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, São Paulo, 2013. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3146/tde-08072014-124306/publico/TESE_LEONARDO_MANZIONE.pdf. Acesso em: mar. 2019.

MARCH, L., STEADMAN, P.**The Geometry of Environment:** an Introduction to Spatial Organization in Design. MIT Press, Cambridge, Massachusetts, 1971.

MARCH, S. T.; SMITH, G. F. **Design and natural science research in Information Technology.** Decision Suport Systems, v. 15, p. 251-266, 1995. http://dx.doi.org/10.1016/0167-9236(94)00041-2

MARSICO, Matheus Lamas; MEDEIROS, Rafael de; DELATORRE, Vivian; COSTELLA, Marcelo Fabiano; JACOSKI, Claudio Alcides. **Aplicação de BIM na compatibilização de projetos de edificações.** Iberoamerican Journal of Industrial Engineering 9 (17), 2017, p. 19-4.

MENGES, Achim; AHLQUIST, Sean. **Computational Design Thinking:** Computation Design Thinking. John Wiley & Sons, 2011.

MILLER, N. **Positioning Computational Designers in your Business** – 4 Things to Consider. 2016. Disponível em: https://provingground.io/2017/02/15/positioning-computationaldesigners-%0Ain-your-business-4-things-to-consider/%0A Acesso em: mar. 2020.

MILLER, Nathan; STASIUK, David. **A Novel Mesh-Based Workflow for Complex Geometry in BIM.** In: ACADIA, New York, 2017, p. 404-413. Disponível em: http://papers.cumincad.org/data/works/att/acadia17_404.pdf. Acesso em: mai. 2019.

MITCHELL, W. J., **Computer-Aided Architectural Design.** Van Nostrand Reinhold, New York, 1977.

MITCHELL, W. J., LIGGET, R. S., KVAN, T., **The Art of Computer Graphics Programming,** VanNostrand Reinhold, New York, 1987

MONTESINOS, Pablo Bellido; GALANT, Fidel Losano; PASCUAL, Francisco. **Experiences learned from an international BIM contest:** software use and information workflow analysis to be published in: Journal of. In: Journal of Building Engineering, Madrid, Spain, 2019. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352710218305680. Acesso em: jun. 2019.

NATIVIDADE, V. G. **Fraturas metodológicas nas arquiteturas digitais.** Dissertação: Mestrado. Universidade de São Paulo. 302 p., 2010.

OXMAN, Rivka. **Theory and design in the first digital age.** Design Studies 27, p. 1–32, 2006.

OXMAN, Vivka. **Digital architecture as a challenge for design pedagogy:** theory, knowledge, models and médium. Design Studies 29, p. 99-120, 2008.

OXMAN, Rivka; OXMAN, Robert. **Theories of the Digital in Architecture**. Routledge, New York, 2014.

OXMAN, Rivka. **Thinking difference:** theories and models of parametric design thinking. Design Studies 52: p. 1–39, 2017.

PROCÓPIO, André *et al.* **Parametria E O Desejo De Uma ComputaçãoIntegrada Em Projeto.** *PIXO - Revista de Arquitetura, Cidade e Contemporaneidade* v. 5, n. 17,

2021.

SEGHIER, T. E.; WAH, L. Y.; AHMAD, M. H.; SAMUEL, W. O. **Building Envelope Thermal Performance Assessment Using Visual Programming and BIM, based on ETTV requirement of Green Mark and GreenRE.** International Journal Of Built Environment And Sustainability, v. 4, n. 3, p. 227–235, 2017.

SILVA, J. L.; MUSSI, A. Q.; SILVA, T. L.; ZARDO, P.; RIBEIRO, L. A. **Desenvolvimento de plug-ins voltados para a análise de requisitos da Norma de Desempenho Brasileira.** GESTÃO & TECNOLOGIA DE PROJETOS, v. 14, p. 46-64, 2019.

SILVA, Lilian; SILVA, Neander; LACROIX, Igor. **Integrating Parametric Modeling with BIM through Generative Programming for the production of NURBS Surfaces and Structures.** In: Proceedings of the 24th CAADRIA Conference - Volume 1, Victoria University of Wellington, Wellington, New Zealand, 15-18, 2019, p. 635-644. Disponível em: http://papers.cumincad.org/data/works/att/caadria2019_103.pdf. Acesso em: jun. 2019.

SOUZA, Edson Eloy. **Arquitetura e geometria.** Arq.urb – Revista eletrônica de Arquitetura e Urbanismo, n.1, p. 105-118, 2008.

SUCCAR, B., 2009. **Building information modelling framework:** a research and delivery foundation for industry stakeholders. Automation in Construction 18 (3), 357–375.

VAISHNAVI, V.; KUECHLER, W. **Design Research in Information Systems.** 2005. Disponível em: <http:// desrist.org/design-research-in-information-systems>. Acesso em: ago. 2021.

VASCONSELOS, Tassia Borges De; SPERLING, David Moreno. **Entre representações, parâmetros e algoritmos:** um panorama do ensino de projeto de arquitetura em ambiente digital na América Latina Among representations, parameters and algorithms: a panorama of digital architectural design teaching in Latin America. n. 2006, 2016.

VELOSO, P. L. A.; SCHEEREN, R.; VASCONSELOS, T. O. **Ensino de projeto e o processo de design paramétrico:** desafios e perspectivas. Nova Hamburgo-RS: Feevale, 2017. p. 17-18.

WOODBURY, R. **Elements of Parametric Design.** Routledge, New York, 2010.

WORTMANN T, TUNÇER B. **Differentiating parametric design:** Digital workflows in contemporary architecture and construction, Design Studies 52, 2017, p. 173-197.

ZEPEDA, Ricardo. **"ArchiCAD ou Revit, qual escolher?" [ArchiCAD versus Revit, ¿cuál elegir?]** 25 Abr. 2019. ArchDaily Brasil. (Trad. Souza, Eduardo). Disponível em: https://www.archdaily.com.br/br/915747/archicad-ou-revit-qual-escolher Acesso em: out. 2021.