XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGENEERING

An educational implementation of the radial-edge data structure

André L. A. Silveira[1], André M. B. Pereira[1], Pedro C. F. Lopes[1], Luiz F. Bez[1], Guilherme Barros[2], Luiz F. Martha[3]

[1]*Institute of Computing, Fluminense Federal University*
Av. Gal. Milton Tavares de Souza, s/n - São Domingos, 24210-310, RJ/Niterói, Brazil
silveira_andre@id.uff.br, andre@ic.uff.br, pedro@ic.uff.br, lfbez@ic.uff.br
[2]*The University of Newcastle, Australia*
University Dr, Callaghan NSW 2308, Austrália
guilherme.barros@newcastle.edu.au
[3]*Department of Civil and Environmental Engineering, Pontifical Catholic University of Rio de Janeiro*
St. Marquês de São Viciente, 225 - Gávea, 22541-041, RJ/ Rio de Janeiro, Brazil
lfm@puc-rio.br

Abstract. This work explores an interactive educational tool that uses the Radial-Edge data structure to simplify the learning of geometric modeling for students and professionals. The Radial-Edge structure was chosen for its robustness in representing complex, non-manifold 3D models, which are common in engineering. However, although the structure is 3D-native, the tool's current implementation focuses on the modeling of 2D solids, establishing a solid foundation for future expansions. The tool translates visual user interactions into operations on the data structure, automating the consistency analysis and topological validation of geometric models. By abstracting the complexity of its parameters, the interface allows the user to focus on modeling concepts and results rather than the underlying topology. This approach improves usability and pedagogical potential, empowering users to efficiently explore and validate models and making advanced geometry manipulation accessible as both a practical resource and an object of study.

Keywords: Solid modeling, Topological data structure, Radial-edge, Python application

1 Introduction

Geometric modeling is a fundamental pillar in several areas of science and engineering, serving as a basis for complex numerical simulations and detailed visual representations. In the context of engineering, for example, the ability to represent and manipulate objects in a virtual environment is crucial for the design, analysis, and optimization of systems and structures. Given this relevance, effective educational tools are essential to empower new professionals, allowing them to intuitively understand and apply these fundamental concepts.

Traditionally, many solid modeling approaches are based on boundary representations (B-Rep), which describe models in terms of their vertices, edges, and faces. Although widely used, 2-manifold representations, where the neighborhood of any point on a surface is homeomorphic to a planar disk, can be limited when representing objects with more complex topologies [1–5]. In such cases, representing objects with multiple regions, such as the intersection of an internal surface with a model's external boundary, requires the ability to handle non-2-manifold geometries [6–8].

In this context, we introduce RETOOL, an educational tool conceived as an evolution of HETOOL [9], expanding its conceptual scope towards 3D modeling. To this end, RETOOL replaces the Half-Edge structure, used by HETOOL for planar representations, with the more robust Radial-Edge structure. It is important to note that although this new structure is designed for complex three-dimensional models, the current implementation of RETOOL deliberately focuses on the modeling of 2D solids. This approach was adopted as a methodological step to first validate the robustness of the implementation in this simpler domain, thereby establishing a verified foundation for future expansion into the 3D environment.

The choice of the Radial-Edge structure is motivated by its proven efficiency in representing models with edges shared by multiple faces, which is inherent to non-2-manifold objects. This allows for a more faithful and versatile representation of complex geometries, such as internal holes, flanges, and non-trivial connections [10, 11]. This tool will allow for the dynamic and interactive manipulation of geometric and topological entities in

a 3D environment.

As an educational platform, RETOOL's design is guided by several key principles. Its architecture seeks to encapsulate the underlying topological complexity, allowing students to focus on high-level geometric modeling concepts instead of the intricate details of the data structure. Furthermore, the tool is designed to be extensible and configurable, which allows for the easy addition of new attributes and functionalities, making it adaptable for different teaching modules or advanced scientific projects. This adaptability is complemented by the ability to integrate with professional simulation environments through the export of models in standardized formats, demonstrating to students how geometric modeling fits into complete engineering and analysis workflows.

With these features, we believe that RETOOL will contribute significantly to the teaching of geometric modeling, offering students, educators, and professionals a powerful and accessible educational resource. The tool aims to tackle the challenges of representing and manipulating complex objects, preparing users to eventually work with advanced topologies, including non-2-manifold models, in their professional practice.

2 Software Description

The geometric modeling library, named RETOOL, was developed in Python, following the paradigms of object-oriented programming. Its purpose is the creation and management of non-2-manifold topological models, utilizing the robust Radial-Edge data structure. This approach allows for the representation of complex geometries, such as multi-volume junctions and internal surfaces, which are essential in advanced engineering and geoscience problems.

2.1 Software Architecture

The RETOOL architecture was designed following the Model-View-Controller (MVC) pattern [12], which promotes a strong decoupling between data representation, visualization, and user interaction logic. This separation results in a more modular and extensible system. The architecture is composed of three main subsystems. The ReModel is the heart of the system, acting as the central data container. It is responsible for storing and managing all topological entities in doubly linked lists (`LinkedList`). The ReView, in turn, functions as a read-only query interface to the ReModel, with the role of translating the topology into simplified geometric primitives for rendering. Finally, the ReController is the application's brain and the sole entry point for model modifications, orchestrating operations, managing the command history (`UndoRedo`) and ensuring system consistency.

The interaction between these components and other support modules, such as computational geometry (`compgeom`) and the attribute manager (`attributes`), can be observed in the architecture diagram in Figure 1.

The fundamental characteristic of RETOOL is its Radial-Edge topological structure. Unlike simpler approaches, it separates the concept of an abstract topological entity from its "use" in a specific context. This conceptual separation is implemented in the `entities_uses.py` file, which defines both the entity classes (Vertex, Edge, Face, Loop, Shell, Region) and the "use" classes (VertexUse, EdgeUse, FaceUse, LoopUse). Classes like VertexUse and EdgeUse represent the instance of an entity in a higher-level context, such as a loop or a shell. The EdgeUse class is the most critical, as it contains the pointers that define the connectivity of the Radial-Edge structure. Among them is the `mate` pointer, which points to the opposite use (with opposite direction) of the same edge; the `radial` pointer, which allows navigation between adjacent faces around an edge; and the `next_in_loop` and `prev_in_loop` pointers, which are used to traverse the circuit of edges that form the boundary of a face. This hierarchy, where topological entities are associated with geometric entities (such as Point, Line, Polyline, and Patch), is illustrated in the class diagram in Figure 2.

To ensure topological consistency during modifications, RETOOL utilizes Euler operators. These are low-level functions that perform atomic changes to the model, such as creating a vertex and a shell (`m_sv`) or creating a face from an existing loop (`m_f`). Each operator is carefully implemented to preserve the validity of the Radial-Edge structure. The inverse operations ("Kill"), such as `k_vs` and `k_ev`, are also provided to safely remove entities.

2.2 Software Functionalities

RETOOL offers a cohesive set of functionalities for interactive geometric modeling, primarily accessible through the ReController.

The creation of geometry and topology is done dynamically. The user can build complex models step-by-

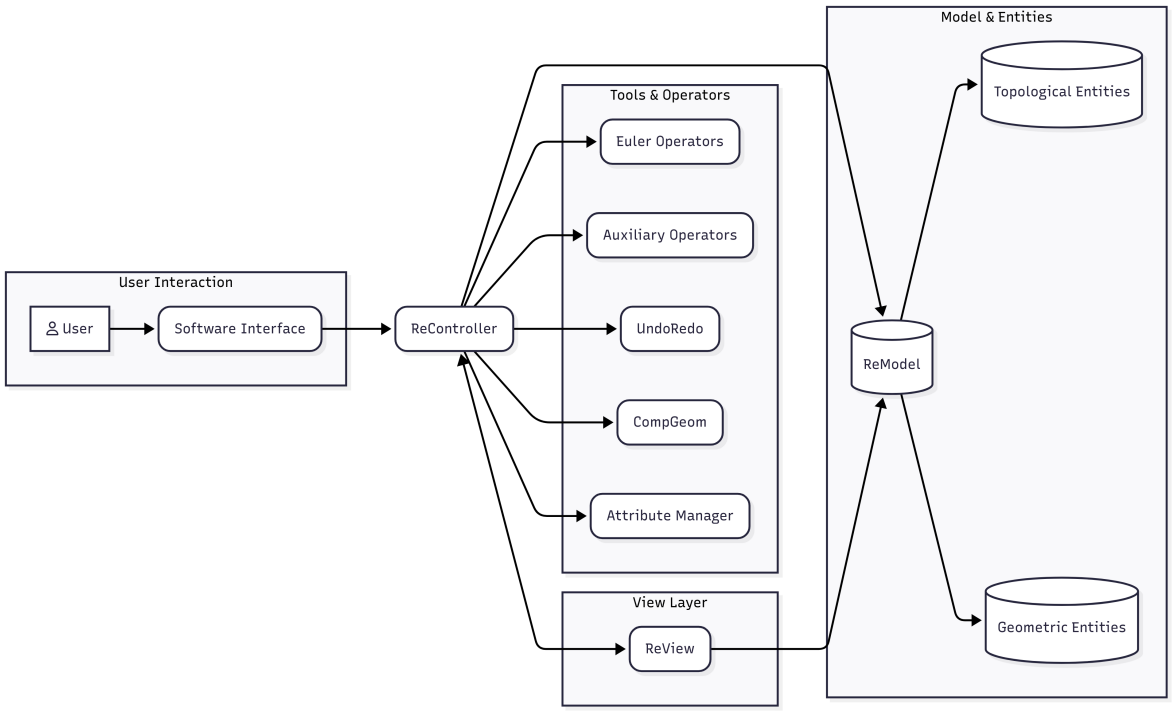


Figure 1. General diagram of RETOOL's modules and directory structure.

step by creating isolated vertices (`createIsolatedVertex`), extruding edges from vertices (`createEdgeFromVertex`) or connecting existing vertices to close contours (`createEdgeBetweenVertices`). From an ordered set of edges that form a closed loop, the library can automatically create a new topological face and its corresponding geometric Patch correctly sewing it to adjacent faces through the `m_f` operator.

One of the most powerful functionalities is the attribute system, which allows semantic properties (such as "Material" or "Boundary Conditions") to be associated with any topological entity. The attribute types are defined in an external `attribprototype.json` file, and their structure is validated by schemas in `attrib_schema.json`, making the system highly configurable and extensible. Additionally, the library can translate these attributes into visual representations, as the `attribsymbols.py` module contains the logic to generate the geometry of symbols, such as arrows for loads and triangles for supports, which can be drawn on the model.

All modification operations performed through the ReController are encapsulated as commands and managed by the `UndoRedo` class. This allows the user to undo and redo their actions, ensuring a non-destructive workflow. The architecture is prepared to save and load models, presumably in a structured format like JSON, allowing for interoperability with other applications.

3 Illustrative examples

This chapter presents three practical examples to demonstrate the application and capabilities of the RETOOL library. The objective is to illustrate how the concepts of the Radial-Edge data structure and the Model-View-Controller architecture, discussed previously, translate into a functional and intuitive geometric modeling tool.

The Figure 3 demonstrates the step-by-step creation process of a geometric model and the respective Euler operators from the RETOOL library called at each stage. The process begins with the creation of a single isolated vertex via the `m_sv` operator. Next, two `m_ev` operations are executed in sequence to create two edges and two new vertices, resulting in an open V-shape.

Subsequently, the structure is closed to form a triangle with the application of an `m_e` operator, which creates the final edge between the existing vertices. The final step illustrates a high-level modeling operation: the insertion of a new vertical edge that starts from the top vertex and intersects the triangle's base. The system resolves this intersection automatically through a sequence of primitive operators (`m_ev`, `k_e`, `m_sv`, `m_e`), which split the original edges and create a new vertex at the intersection point to maintain the model's topological consistency. This example illustrates how the library allows for the construction of complex geometries through a sequence of fundamental operators.

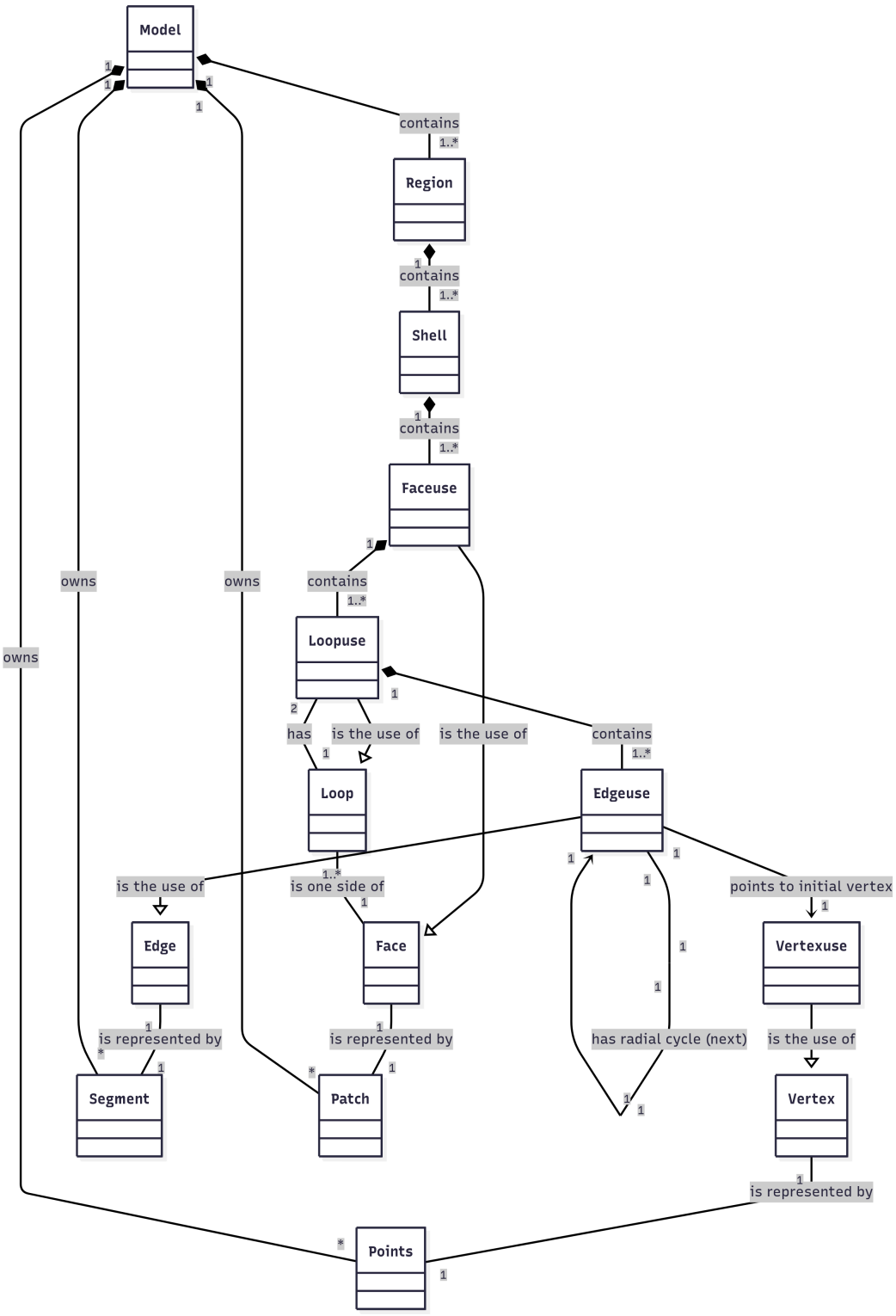


Figure 2. Class diagram of RETOOL, showing the relationship between topological entities, their “uses” and the corresponding geometric entities.

Figure 4 presents a simple geometric model created with the RETOOL library, demonstrating a basic planar subdivision. The model consists of a single triangular face, described using the Boundary Representation (B-Rep) approach in terms of its topological entities: Vertices (black dots), Edges (blue lines), and the Face (grey area). A central feature of the library, illustrated in this example, is the automatic recognition and generation of the planar face at the moment a connected loop of edges is closed, creating a region.

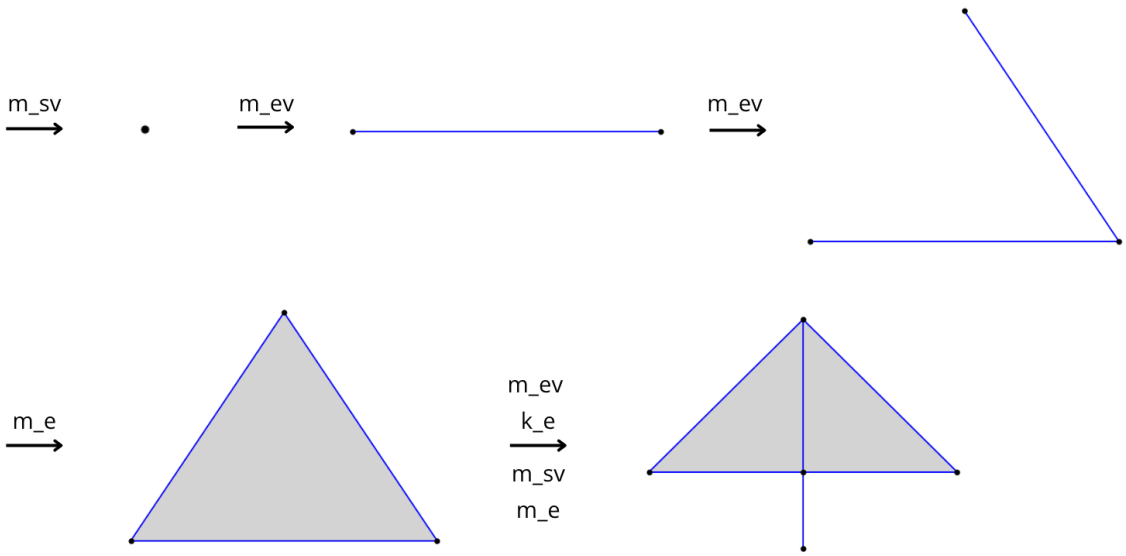


Figure 3. Step-by-step modeling process in the Retool library, showing the sequence of Euler operators to build a geometry and resolve an edge intersection.

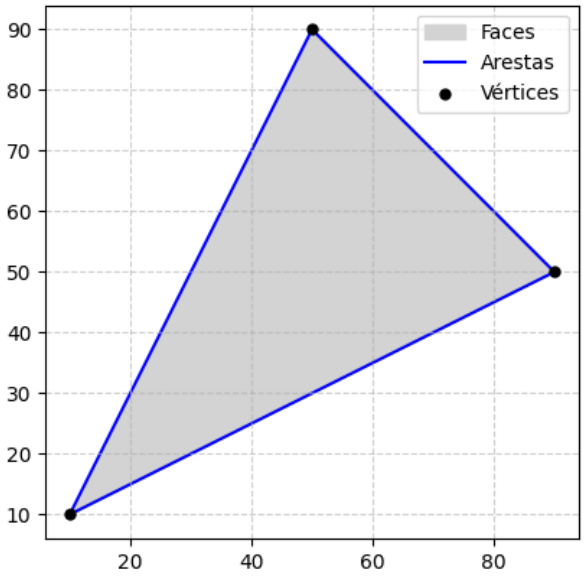


Figure 4. A simple B-Rep model of a triangular face, showing its constituent topological entities (Vertices, Edges, and Face).

A third example involving the creation of a rectangular beam composed of two adjacent faces. The model is built through the 'ReController', where the library demonstrates its capability to automatically generate faces as soon as an edge contour is closed. Subsequently, the model is semantically enriched by applying various engineering attributes, including assigning distinct materials to each face, imposing different support conditions on the left vertices (restricting displacements in one or both axes), and applying a concentrated load. The final visualization, presented in Figure 5, displays not only the geometry but also the graphical symbols and colors corresponding to these attributes, illustrating the complete cycle from topological modeling to the representation of engineering data.

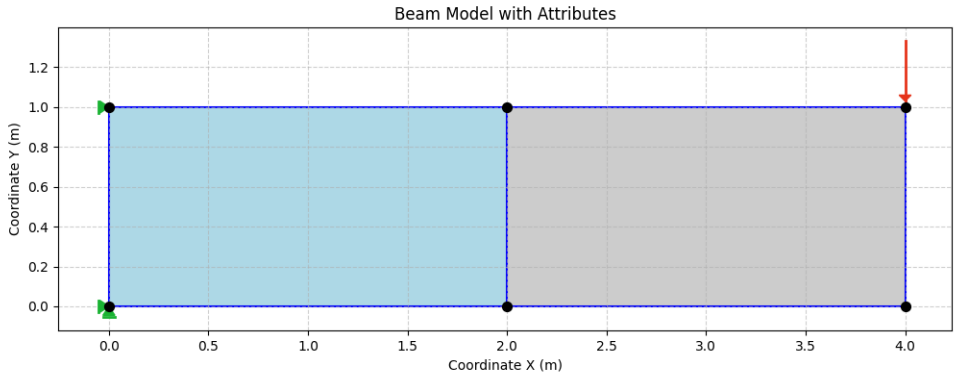


Figure 5. A beam model created in RETOOL, showing the application of attributes: two different materials (colors), support conditions on the left, and a concentrated load on the top-right.

4 Conclusions

This work presented RETOOL, an open-source library for geometric modeling that employs a data structure based on the Radial-Edge. Although the Radial-Edge structure is robust for representing complex, non-manifold 3D models, RETOOL's current implementation deliberately focuses on the modeling of 2D solids, establishing a solid foundation for future expansions. Developed with a Model-View-Controller (MVC) architecture, the package allows for the creation and management of complex topological entities in an interactive and visual manner. RETOOL was conceived as an educational tool, abstracting the underlying complexity so that the user can focus on modeling concepts.

The library also allows for the configuration of elements by assigning semantic properties, such as "Material" or "Boundary Conditions," in an extensible way through JSON files. All modification operations are encapsulated and controlled by an undo/redo system, which ensures a non-destructive workflow. Additionally, the architecture is prepared to save and load the created models, allowing for interoperability with other applications.

In the future, it is expected to enhance this library, expanding its implementation to the 3D environment and consolidating its use as both an educational tool and a practical resource for the analysis of engineering problems.

Acknowledgements. This research was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), grant no. 308884/2021-3 and grant no. 140639/2024-1, and the Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), grant no. E-26/201.224/2022.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] J. Qian, X. Luo, and Y. Xue. Half-edge composite structure: good performance in motion matching. *Journal of Ambient Intelligence and Humanized Computing*, vol. , 2022.

[2] Y. Zhang, X. Luo, and J. Jia. A compact face-based topological data structure for triangle mesh representation. *Comput-Aided Des Appl*, vol. 16(3), pp. 539–557, 2019.

[3] H. Karim, A. Abdul Rahman, P. Boguslawski, M. Meijers, and van P. Oosterom. *The Potential of the 3D Dual Half-Edge (DHE) Data Structure for Integrated 2D-Space and Scale Modelling: A Review*, pp. 477–493. Springer International Publishing, Cham, 2017.

[4] P. Boguslawski and C. Gold. Buildings and terrain unified – multidimensional dual data structure for gis. *Geo-spatial Information Science*, vol. 18, n. 4, pp. 151–158, 2015.

[5] P. Boguslawski and C. Gold. The dual half-edge—a topological primal/dual data structure and construction operators for modelling and manipulating cell complexes. *ISPRS International Journal of Geo-Information*, vol. 5, n. 2, 2016.
[6] A. Chatzivasileiadi, N. M. Wardhana, W. Jabi, R. Aish, and S. Lannon. Characteristics of 3d solid modeling software libraries for non-manifold modeling. *Comput-Aided Des Appl*, vol. 16(3), pp. 496–518, 2019.
[7] W. Jabi. Linking design and simulation using non-manifold topology. *Architectural Science Review*, vol. 59, n. 4, pp. 323–334, 2016.
[8] V. Dyedov, N. Ray, D. Einstein, X. Jiao, and T. J. Tautges. Ahf: array-based half-facet data structure for mixed-dimensional and non-manifold meshes. *Engineering with Computers*, vol. 31, n. 3, pp. 389–404, 2015.
[9] D. S. Bomfim, R. L. Soares, L. F. Bez, P. C. Lopes, A. M. Pereira, and L. F. Martha. Hetool: A half-edge topological object-oriented library for generic 2-d geometric modeling. *SoftwareX*, vol. 21, pp. 101307, 2023.
[10] K. J. Weiler. *Topologial structures for geometric modeling*. PhD thesis, Faculty of Rensselaer Polytechnic Institute, Troy, New York, 1986.
[11] L. F. C. R. Martha. *Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three-Dimensions*. PhD thesis, Faculty of the Graduate School of Cornell University, Ithaca, New York, 1989.
[12] G. E. Krasner and S. T. Pope. A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80. *The Journal of Object-Oriented Programming*, vol. 1, n. 3, pp. 26–49, 1988.