2026 European Conference on Computing in Construction
Corfu, Greece
July 12–15, 2026

SPACE-DRIVEN KNOWLEDGE: A TOPOLOGICAL FRAMEWORK FOR SEMANTIC INFERENCE IN BUILDINGS

Isabelle Fitkau, Max Hammer, and Maximilian Sternal
Technische Universität Berlin, Germany

Abstract

Current building modeling approaches treat geometry as primary and semantics as secondary, limiting knowledge-integrated building design where spatial reasoning informs iterative design decisions. This paper presents a space-driven knowledge framework that integrates topological space partitioning with semantic triple generation, enabling immediate inference over spatial relationships. The workflow progresses through iterative stages of partitioning, algorithmic triple generation from topological queries and quantity calculations, axiomatic reasoning, and rule-based inference for requirements. A proof of concept demonstrates how storey classification, spatial adjacency, and fire safety requirements can be inferred during the modeling process using the BOT ontology extended with fire safety concepts from FiSa.

Introduction

Building design is inherently iterative (Pauwels and Bod, 2014). Regulatory requirements arise from building-level classifications to component-level specifications, depending on geometric and functional properties that emerge during design. However, current workflows separate modeling from compliance checking, meaning designers receive feedback on regulatory implications only after committing to spatial configurations.

This paper argues for a different approach in which semantic knowledge is assigned simultaneously with geometric modeling, allowing immediate reasoning about classifications and requirements. Rather than treating semantics as metadata attached after modeling geometry, we propose treating every modeling operation also as a knowledge-modeling event. When a designer partitions a building envelope into storeys, this operation should not only create geometric objects, but also instantiate semantic resources, compute quantities, and trigger inference rules that classify those objects.

The technical foundation combines two domains. Topology-aware modeling kernels, such as Topologic (Jabi and Chatzivasileiadi, 2021) and the Partition Platform (Sternal et al., 2025; Sternal, 2025; Huhnt et al., 2023), provide space partitioning operations that inherently compute spatial relationships. Semantic web technologies such as triple stores, ontologies, SPARQL, and rules provide mechanisms for representing, querying, and reasoning over knowledge. This paper presents a framework coupling these into a ”space-driven knowledge” workflow: a topology-aware kernel performing space partitioning, a triple store persisting semantic knowledge using BOT (Rasmussen et al., 2017) and FiSa (Fitkau and Hartmann, 2024), and an inference engine deriving new knowledge. The workflow proceeds through iterative cycles of **model**, **assign**, **compute**, and **infer**. Triples are generated algorithmically from topological queries, axiomatically through OWL reasoning, or through custom rules.

With spatial relationships and derived classifications explicitly represented in real-time throughout the architectural design process, changes to geometry automatically derive to updated inferences, enabling exploration of alternatives, e.g., with awareness of their regulatory consequences. Recent frameworks for the evaluation of early-phase design variants integrate spatial, structural, and environmental feedback in iterative design loops (Napps et al., 2026). Our work complements such approaches by providing the topological and semantic basis from which variant-specific classifications and requirements can be derived continuously during modeling. This paper makes two contributions: (1) a framework coupling geometric and topological modeling with semantic inference, and (2) a proof of concept demonstrating storey classification and fire safety requirement inference. We focus on the building scale while indicating how the approach extends to multi-scale contexts.

Background

First, a theoretical background introduces the main concepts of the proposed design workflow.

Design as an Iterative Process

Current CAD and BIM tools support geometric iteration and semantic attribute management, but evaluative knowledge remains disconnected from the modeling process. Requirements checking typically occurs after design decisions are made (Eastman et al., 2009; Pauwels et al., 2011), despite design being an inherently iterative, model-based reasoning process (Pauwels and Bod, 2014).

Coupling semantic inference with geometric modeling allows designers to observe how spatial configurations affect classifications and requirements during iteration (Pauwels et al., 2024). This requires information systems in which

spatial operations produce both geometric objects and semantic assertions.

Space Partitioning

Traditional geometric modeling represents shapes through boundary representations (B-rep) or constructive solid geometry (CSG), focusing on the geometry itself rather than its spatial relationships. Topology-aware modeling approaches instead emphasize the relational structure of space through the use of topologically connected data structures, creating a consistent hierarchical partition of space through subdivision (Huhnt et al., 2023). Therefore, such data structures offer direct topological queries without geometric computations (Massafra et al., 2024). However, these relationships typically remain within the geometric representation rather than being exposed as querable semantic triples (Donkers and Petrova, 2025). It should be noted that the modeling operations employed in topology-aware kernels, e.g., extrusion, slicing, and placement of primitives, are not fundamentally different from the established parametric design gestures in current CAD and BIM tools (Sternal et al., 2025).

The key distinction is not in the operators themselves but in the underlying paradigm. Many tools follow an object-centric approach, constructing components individually and validating consistency post-hoc. Space partitioning follows a space-centric approach in which the entirety of Euclidean space is subdivided into non-overlapping domains with inherently consistent topology. This guaranties manifold consistency by construction and provides a reliable geometric model for the semantic enrichment proposed in this paper.

The idea of a space-centric approach dates back to the 1970s, when soap-bubble and grid-bubble space management systems were explored (Eastman, 1999), yet adoption declined in favor of component-based tools that prioritize individual building elements over spatial coherence. Our framework addresses this gap by synchronizing topological computations with semantic knowledge graphs, going beyond what both early space management systems and current component-based tools offer.

Knowledge Management in Building Modeling

Building information workflows typically separate geometric modeling from semantic enrichment. This separation means that feedback on regulatory implications or other design related decision support arrives after spatial configurations are already established (Pauwels et al., 2024). Industry Foundation Classes (IFC) provide a unified data model that integrates geometry and semantics. However, file-based exchange treats semantic assignment as a pre-export step rather than a part of the modeling process itself. Authoring tools default to generic element types, and adding semantic knowledge requires effort disconnected from geometric manipulation (Beetz et al., 2009).

Semantic web technologies enable multiple domain-specific descriptions to coexist within a linked web of data, motivating coupling between modeling environments and semantic representations (Pauwels et al., 2012). Linked Building Data approaches represent building information as semantic graphs using the Resource Description Framework (RDF), which allows SPARQL queries and reasoning (Pauwels et al., 2011). Lightweight domain ontologies, such as the Building Topology Ontology (BOT) (Rasmussen et al., 2017), provide vocabularies for spatial structures corresponding to how designers progressively subdivide space. These representations can participate in design iterations when coupled with modeling environments.

Triple stores persist RDF data and support SPARQL queries. The deployment ranges from embedded libraries (RDFLib, RDF4J, Apache Jena) as shown by Elshani et al. (2026), to server-based endpoints (Fuseki, GraphDB) and federated architectures that distribute data between stakeholders (Werbrouck et al., 2024). Embedded approaches reduce latency for design-time integration, while federated architectures support collaboration but introduce consistency challenges during iteration.

Whereas some CAD tools are limited by restricted programmatic access to geometric and topological data through proprietary APIs, custom topology-aware kernels compute adjacency, containment, and connectivity as by-products of partitioning. When a horizontal plane subdivides a building volume into storeys, the kernel reports the resulting zones and which zones share faces or contain others. These facts map directly to BOT relationships (`bot:adjacentZone`, `bot:containsZone`), enabling triple generation from geometric operations. This provides a foundation for inference-guided design: the topology-aware kernel produces spatial relationships, the semantic model assigns and persists triples, and the inference engine derives classifications and properties. Computed quantities and extracted relationships feed classification rules that derive regulatory implications while geometry remains open to change. At the same time, data remain in their originating source.

Ontologies and Inference

Ontologies formalize domain knowledge as concept hierarchies and relationships interpretable by machines (Gruber, 1993). RDF represents knowledge as graphs of subject-predicate-object statements. OWL extends RDF with description logic semantics (Pauwels et al., 2011), which supports automated reasoning. Class membership can be inferred from property values, and property characteristics (transitivity, symmetry, inverse) derive knowledge through the graph. However, using OWL does not require utilizing its reasoning capabilities. Leveraging inference requires triggering a reasoner (Pellet, HermiT) that evaluates axioms against asserted facts. When reasoning is active, two mechanisms derive knowledge:

Axiomatic reasoning derives class memberships through necessary and sufficient conditions. OWL provides restrictions that constrain what holds for instances of a class.

Reasoners evaluate these definitions and automatically assign instances to their respective classes. This suits classifications derivable from properties, such as storey type from height and containment from transitivity. The same restrictions that enable classification also support consistency checking: a reasoner can detect when asserted facts contradict class definitions.

Rule-based inference handles conditions beyond OWL's expressivity, particularly arithmetic comparisons and conditions spanning multiple objects and properties. Custom rules can be written in rule languages, such as SWRL (Horrocks et al., 2004), Jena's rule syntax, Prolog, or native code. The mechanism often matters more than the syntax. When antecedent conditions match, consequent triples are asserted. Work on automated code checking (Eastman et al., 2009; Pauwels et al., 2011) demonstrated that regulatory logic can be encoded as executable rules in various formalisms.

We want to incorporate semantic inferences directly into the design process. We are not arguing against the continued use of rule-based applications for automatic compliance checking. However, an important distinction concerns missing information. OWL operates under the Open World Assumption (OWA), where absent statements are unknown, not false. Rather than lacking a fire stair, a building without an asserted fire stair might simply lack that information. OWL restrictions enable classification and consistency checking but cannot verify completeness. Constraint languages like SHACL operate under the Closed World Assumption (CWA), where absent information indicates a violation. Shapes verify that required properties exist and satisfy cardinality or value constraints, similar to OWL restrictions, but with different semantics suited for validation rather than classification. For iterative design, the OWA inference derives classifications and requirements from available information without treating incompleteness as an error. CWA validation fits later stages when completeness is expected before submission and permitting processes.

Our proposed framework uses OWL axioms and restrictions for classification and consistency, as well as custom rules for requirement property inference.

Method

The framework couples three components (Figure 1): a topology-aware modeling kernel, an RDF triple store, and an inference engine. Each component has distinct responsibilities and user inputs in the triple generation process.

Topology-Aware Modeling Kernel

The modeling kernel handles geometric operations: space partitioning through parametric design operations (extrude, slice, subdivide) and geometric checks. It maintains topological relationships between cells, including which faces are shared, which cells are adjacent, and which cells contain others.

The kernel exposes these relationships through a query interface. When partitioning creates new cells, the kernel reports not only the resulting geometry but also the topological facts: *Cell A and Cell B share Face F and are therefore adjacent,* which feeds the semantic model in the triple store. Example technologies include Topologic, which provides cellular complex operations via Python/C++ for non-manifold modeling, and the Partition Platform, which offers stepwise partitioning with history tracking via a custom Java kernel for manifold modeling. The presented framework is kernel-agnostic: any system that provides partitioning operations with topological relationship extraction can serve this role.

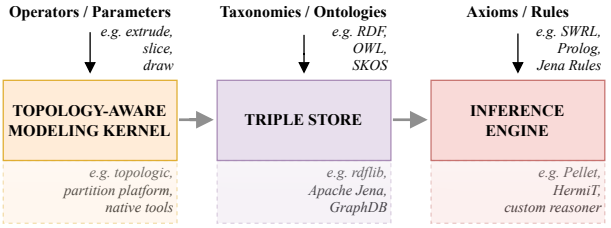

Figure 1: Framework architecture: topology-aware modeling kernel, triple store, and inference engine with their respective inputs and example technologies.

Triple Store

The triple store persists the semantic model as RDF graphs and provides query access to the knowledge base. It serves as the central repository where modeling operations generate triples, spatial relationships are stored, and inferred knowledge is materialized and maintained. RDF represents knowledge as serializable subject-predicate-object statements. Triple stores enable SPARQL queries over persisted graphs, allowing for the retrieval of objects, traversal of relationships, and filtering by properties. Updates to the triple store are managed through an inference model using a reasoner. This way, updates remain synchronized with the underlying storage, coordinating both asserted and inferred triples in their respective graphs.

Multiple named graphs can separate asserted facts (user-specified and algorithmically generated) from inferred knowledge for provenance tracking and inference validation. Implementation options range from embedded libraries to server-based systems. Embedded approaches minimize latency for design-time operations, while server-based deployments enable multi-user access and federated architectures (Werbrouck et al., 2024); however, synchronization complexity increases during iterative modeling.

Inference Engine

Inference engines derive new knowledge from asserted facts by applying logical rules and axioms over RDF triples (Eastman et al., 2009; Pauwels et al., 2011). Deductive reasoners apply description logic axioms to classify instances and derive properties through characteristics such as transitivity, symmetry, and inverse relationships. While axioms work well for property-based classifications, rule-based engines cover conditions beyond the expressiveness

of description logic. Forward chaining materializes all derivable triples upfront, while backward chaining can derive them on demand.

In building modeling, inference engines enable automated regulatory compliance checking, where geometric changes trigger the re-evaluation of derived classifications and requirements (Pauwels et al., 2024). This enables the exploration of regulatory consequences during design iterations. Domain ontologies such as ifcOWL (Beetz et al., 2009), BOT, and FiSa encode spatial relationships and component specifications that inference engines can operate on incrementally (triggered by triple store updates) or on-demand (invoked before queries). Incremental inference maintains a consistent knowledge state but incurs overhead with every update, whereas on-demand inference delays computation but may cause query latency.

Knowledge Derivation Mechanisms

Triple generation occurs through three distinct mechanisms, each suited to different knowledge derivation patterns.

Algorithmic generation produces triples directly from modeling operations and topological queries. Space partitioning operations enable object type assignment and hierarchical instantiation. Topological queries, e.g., extracting adjacency from shared faces, and geometric computations yield quantitative properties from spatial dimensions and positions. This mechanism handles knowledge derivable through direct computation on geometry and topology without requiring logical inference.

Axiomatic reasoning leverages description logic semantics to derive class membership. Reasoners evaluate class restrictions and property characteristics to materialize implicit knowledge. Classification emerges through restriction-based reasoning when instances satisfy class definitions.

Rule-based inference addresses conditions that exceeds the expressiveness of description logic. Custom rules encode arithmetic comparisons, multi-object properties, and regulatory threshold evaluation. This mechanism can handle procedural knowledge and numeric reasoning.

The framework's effectiveness derives from this separation, preserving each layer's responsibilities while leveraging the strengths of each formalism.

Space-Driven Knowledge Design Workflow

The workflow proceeds through iterative cycles, progressing to finer levels of detail (Figure 2). Architectural design development follows different spatial strategies: subdivision decomposes larger volumes into smaller ones, while aggregation composes spaces from individual units. The workflow presented here follows a subdivision strategy, progressively partitioning space from coarse to fine granularity. While aggregation-based and hybrid strategies exist in design practice, subdivision aligns naturally with the space partitioning paradigm, where the entirety of space is recursively divided into non-overlapping domains. At each stage, four activities occur: modeling (geometric partitioning), semantics (type assignment), compute (quantity extraction and topological queries), and inference (classification and reasoning). Each geometric cell created through partitioning receives a unique URI, initially asserting minimal type information. Semantic assignment then specifies object types, while computed quantities and queried topological relationships enrich the object's semantic description through subsequent workflow stages. As properties accumulate through stages, more rules apply, and further knowledge becomes inferable. The following description uses the BOT ontology extended by FiSa concepts, although this approach transfers to other ontology-based knowledge formalizations.

Stage 1: Partitioning by Terrain

The workflow begins with site context. Partitioning space into *'air'* and *'earth'* establishes the ground plane and terrain geometry. Semantically, this creates a `bot:Site` instance. Extracted quantities include terrain height, geographic coordinates, parcel boundaries, as well as relationships to adjacent zones. Inference at this stage determines the applicable regulatory framework based on geographic location through jurisdiction-specific variants, establishing the assessment basis for subsequent classifications.

Stage 2: Partitioning by Building

Within the site, further subdivision partitions 'air and earth' into building volumes. A building becomes a `bot:Building` instance with the sites containment relationship (`bot:hasBuilding`). Adjacent sites establish `bot:adjacentZone` relationships relevant to regulatory requirements. Computed quantities include gross floor area and the number of utilization units. Topological queries determine whether the building is freestanding (no adjacencies to other buildings), while geometric computation can further derive whether special building conditions prevail, such as sales premises or high-rise building classification.

Stage 3: Partitioning by Storey

Horizontal planes partition buildings into storeys. Each storey is instantiated as `bot:Storey` with hierarchical relationships (`bot:hasStorey`). Semantic assignment specifies usage types (`fisa:hasUsage`) and further defines utilization units within the building hierarchy. Computed quantities extract ceiling heights (`fisa:hasUpperCeilingHeight`) and average floor elevations relative to ground (`fisa:hasAverageHeight`). Queries determine the vertical position within the building and count the residential units per storey. These quantities enable storey classification through axiomatic reasoning, which derive storeys as `fisa:AboveGroundStorey` or `fisa:Basement` based on height and elevation properties, and further sub-classification into `fisa:UpperStorey` and `fisa:GroundFloor` according to position. Building class inference also occurs at this stage through rule-

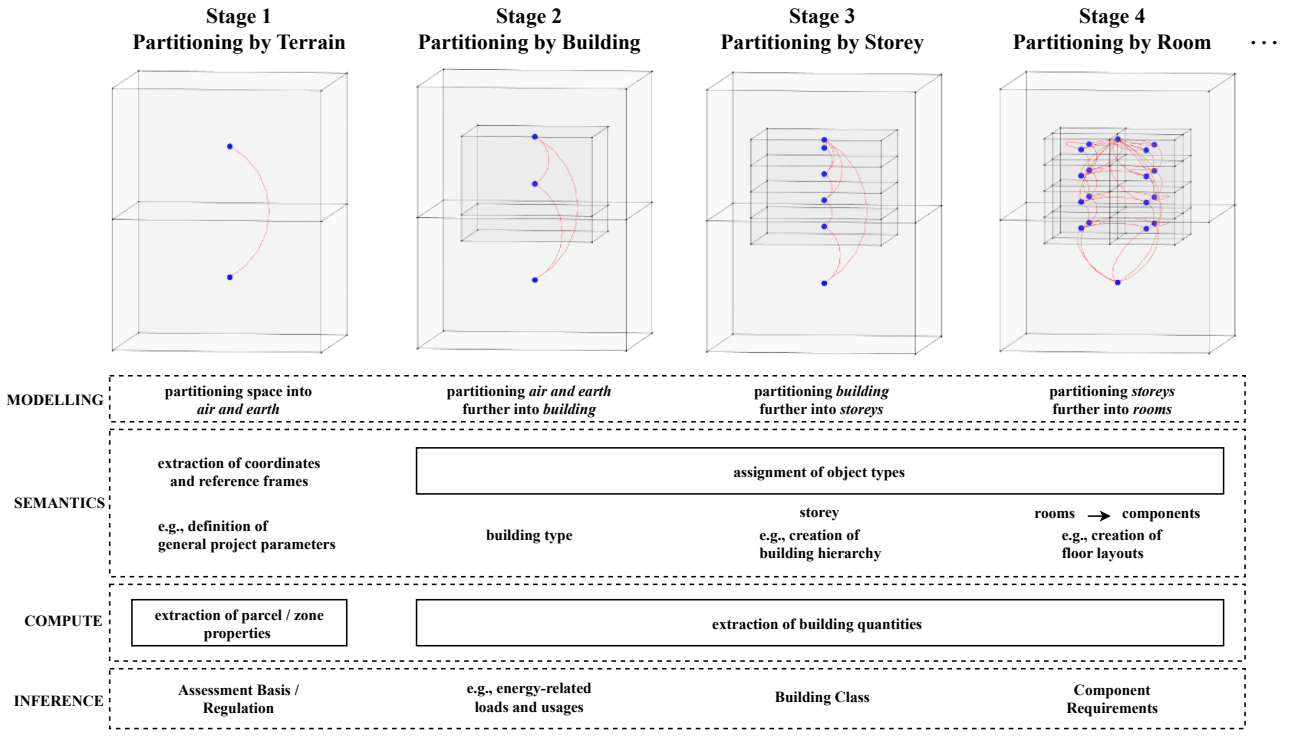

Figure 2: Iterative workflow through four stages of partitioning. Each stage involves geometric modeling, semantic assignment, quantity computation and topology querying, and inference, with results feeding back into subsequent design iterations.

based reasoning. Rules evaluate accumulated properties, such as building height, freestanding status, number of utilization units, and gross floor area, to derive a `fisa:BuildingClass`. This class inference determines component fire resistance and behavior requirements applied in subsequent stages.

Stage 4: Partitioning by Room

Further partitioning operations subdivide storeys into spaces, here also termed rooms. Each space is assigned a `bot:Space` instance. Walls separating rooms instantiate as `fisa:Wall` elements with relationships to adjacent spaces (`bot:adjacentElement`). Semantic assignment specifies space usage, necessary stairs, corridors, and other relevant types. Topological queries directly instantiate semantic knowledge, such as detecting faces with adjacency to outside zones identifies external walls (`fisa:isExternal`), while containment analysis conclude space-enclosing components (`fisa:isSpaceEnclosing`). Inference at this stage derives component requirements. Based on the building class (inferred in Stage 3), component position and properties (external, load-bearing), and adjacency relationships, rules infer the required fire resistance (`fisa:hasRequirementOfFireResistance`) and fire behavior (`fisa:hasRequirementOfFireBehaviour`) for each component.

Iterative Design Refinement

The workflow operates through feedback loops. Geometric modifications trigger quantity recomputation, which affects requirement inference. For example, when storey height changes, the system updates derived properties, re-evaluates the building class, and derives component requirements accordingly. The iterative stages, from geometry change through quantity update to classification inference and requirement derivation, provide feedback on the regulatory consequences of spatial decisions, enabling informed exploration of design alternatives.

Proof of Concept

A proof-of-concept implementation demonstrates the framework's space-driven knowledge generation capabilities. The implementation generates triples from spatial operations for space classification, as well as regulatory requirement inference at the component-level.

Implementation Overview

The Partition Platform (Sternal, 2025) used in this proof of concept serves as the topology-aware modeling kernel, providing stepwise space partitioning from coarse (air/earth, parcel, building envelope) to fine (storeys, rooms, walls) granularity. Each partitioning operation subdivides geometric cells and stores their topological relationships, e.g., the adjacency of each cell through shared faces and containment through hierarchical nesting. The platform's GUI is built using JavaFX, enabling interactive visualization and manipulation of the partitioning process. The semantic model builds on BOT for spatial structure and extends it with FiSa for fire safety concepts. The implementation uses Apache Jena 5.6.0 for RDF graph management and SPARQL querying, with Openllet 2.6.5 pro-

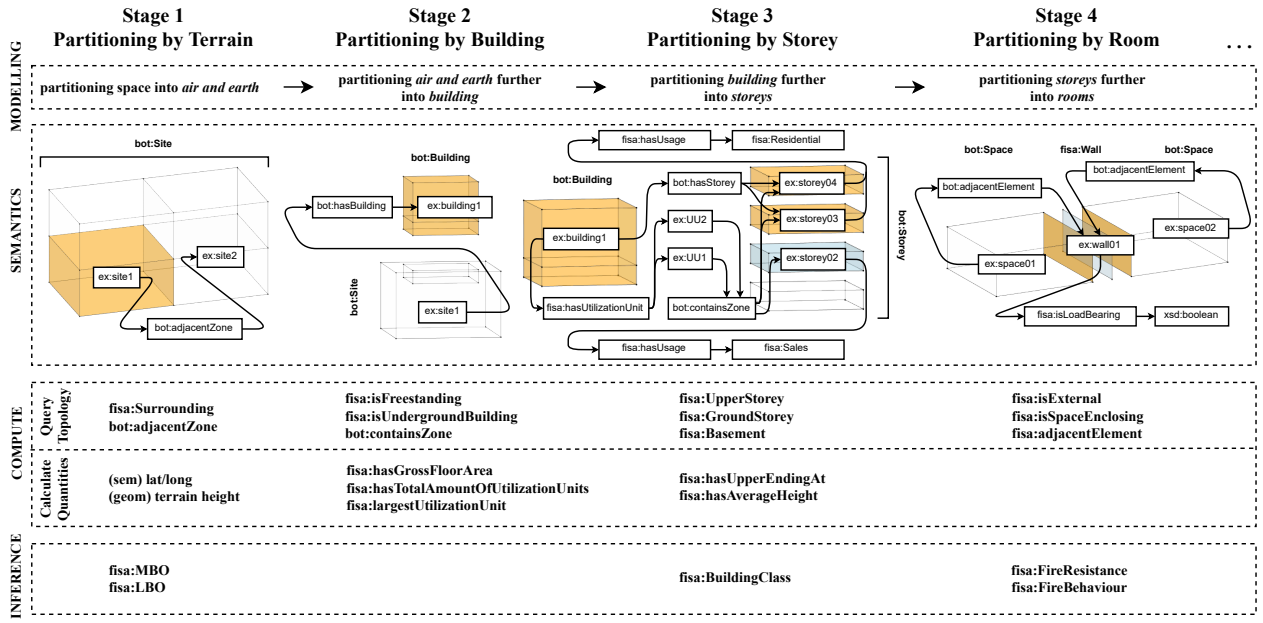

Figure 3: Proof of concept demonstrating semantic triple generation at each partitioning stage, from site context through room subdivision, with corresponding inferences.

viding OWL reasoning capabilities. Algorithmic triple generation translates partitioning operations into object instantiations and BOT object properties directly in Jena's TDB2 Triple Store. Axiomatic reasoning derives classifications from geometric and topological properties. The rule-based inference engine Openllet derives knowledge written in custom SWRL rules within FiSa. Notably, the ontology was verified using the Pellet reasoner during development. The custom inference engine used in this proof of concept only implements the reasoning features required for the present scope, namely transitive property characteristics and SWRL rule execution. Apache Jena's InfModel coordinates the reasoning process, managing the interaction between the TDB2 triple store and the inference engine while maintaining synchronization between asserted and inferred triples.

Demonstration Cases

The following three cases demonstrate the triple generation process across successive partitioning stages, from storey classification (Stage 3) through spatial adjacency to space classifications and component-level inference of fire resistance requirements (Stage 4). Each case illustrates a different source of semantic enrichment, such as topological queries derived from the modeling kernel, computed properties from geometric analysis, and inferred properties from ontology reasoning (see Figure 3).

Case 1: Classification

When a storey is created by partitioning a building in the geometric model and asserted as a `bot:Storey`, properties are automatically derived from geometric analysis and topological queries, triggering classification within the FiSa storey hierarchy. As shown in Figure 4, `Space_01` is automatically classified as `fisa:AboveGroundStorey`. This classification follows from FiSa's axiom defining `fisa:AboveGroundStorey` as storeys whose floor surface has no earth contact and lies at a certain height above terrain height.

▸ Run Query | Refresh | Auto-refresh (5s) | Preset Queries...

subject	subjectName	property	object	objectName
obj:c3d4e5f6-...	Storey_01	rdf:type	class:SemanticObject	
obj:c3d4e5f6-...	Storey_01	rdf:type	bot:Storey	
obj:c3d4e5f6-...	Storey_01	rdf:type	fisa:AboveGroundStorey	
obj:c3d4e5f6-...	Storey_01	rdf:type	owl:NamedIndividual	
obj:c3d4e5f6-...	Storey_01	rdf:type	bot:Zone	
obj:c3d4e5f6-...	Storey_01	rdf:type	owl:Thing	
obj:c3d4e5f6-...	Storey_01	bot:hasSpace	obj:d4e5f678-9012-3456-7890-abcdef123456	Space_01
obj:c3d4e5f6-...	Storey_01	bot:hasSpace	obj:e5f67890-1234-5678-90ab-cdef12345678	Space_02
obj:c3d4e5f6-...	Storey_01	bot:hasSpace	obj:f6789012-3456-7890-abcd-ef1234567890	Space_03
obj:c3d4e5f6-...	Storey_01	bot:hasElement	obj:a1b2c3d4-e5f6-7890-1234-56789abcdef0	Wall_01_internal
obj:c3d4e5f6-...	Storey_01	bot:hasElement	obj:38d9ebf3-5f8a-401b-821a-564e3e957251	Wall_02_internal
obj:c3d4e5f6-...	Storey_01	bot:hasElement	obj:0a475dc6-400f-42a3-a58e-7a9cdccfcc75	Wall_03_external
obj:c3d4e5f6-...	Storey_01	bot:hasElement	obj:28410c91-d0a0-4696-9879-555136bd0ae4	Wall_04_external
obj:c3d4e5f6-...	Storey_01	bot:containsZone	obj:d4e5f678-9012-3456-7890-abcdef123456	Space_01
obj:c3d4e5f6-...	Storey_01	bot:containsZone	obj:e5f67890-1234-5678-90ab-cdef12345678	Space_02
obj:c3d4e5f6-...	Storey_01	bot:containsZone	obj:f6789012-3456-7890-abcd-ef1234567890	Space_03

Results: 16 rows

*Figure 4: Results show relationships for `Storey_01` for a SPARQL-Query for storeys in custom Database Viewer. (**purple**: user-asserted; **orange**: computed in kernel; **red**: inferred; **green**: `Space_01`)*

Case 2: Adjacency

Spatial adjacency relationships are derived by topological queries. The kernel computes `bot:adjacentZone` relations between spaces sharing boundaries, as shown for `Space_01`'s connections to `Space_02` and `Space_03`, as well as their `bot:adjacentElement` relations to the bounding walls (see Figure 5).

Case 3: Component Requirements

Fire resistance requirements are derived from custom rules. When a wall is asserted with boolean datatypes in both `fisa:isLoadBearing` and

Listing 1: Sparql-Query for Spaces and their inferred triples

```
PREFIX bot: <https://w3id.org/bot#>
PREFIX prop: <http://bitub.de/semantic/property/>

SELECT ?subject ?subjectName ?property ?object ?
    objectName
WHERE {
 GRAPH <http://bitub.de/semantic/inferred> {
   ?subject a bot:Space.
   ?subject ?property ?object .
 }
 OPTIONAL {?subject prop:name ?subjectName .}
 OPTIONAL {?object prop:name ?objectName .}
}
```

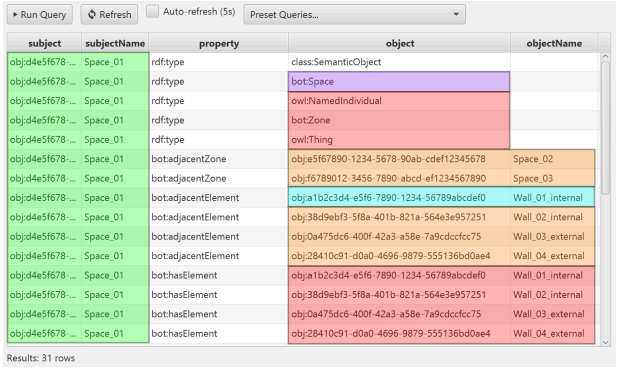

*Figure 5: Results show relationships for `Space_01` for a SPARQL Query for spaces (Listing 1) in custom Database Viewer (**purple**: user-asserted; **orange**: computed in kernel; **red**: inferred; **green**: `Space_01`; **lightblue**: `Wall_01`)*

`fisa:isSpaceEnclosing`, those rules infer the object property `fisa:hasRequirementOfFireResistance`. Consequently, `Wall_01_internal` is inferred to require `fisa:feuerhemmend` (Figure 6), following regulatory requirements for load-bearing internal walls in the considered spatial configuration.

Discussion

This paper presents a framework that couples topology-aware modeling with semantic triple generation, classification and inference, enabling real-time feedback on the regulatory implications of geometric decisions. The proof of concept demonstrates that models can be semantically enriched during design stages through the combination of topological queries, axiomatic reasoning, and custom rules.

The current implementation demonstrates feasibility rather than production readiness. Several technical limitations remain unaddressed. The modeling kernel handles straightforward partitioning operations but does not address complex parametric operations or a merging of existing geometric models. Performance at scale, involving thousands of spaces and millions of triples, remains unevaluated. The proof-of-concept implementation operates on a JavaFX 21 UI for the Database View and the Partition Platform, utilizing Apache Jena's TDB2 triple store and the Openllet reasoner, rather than being a fully integrated or web-based application.

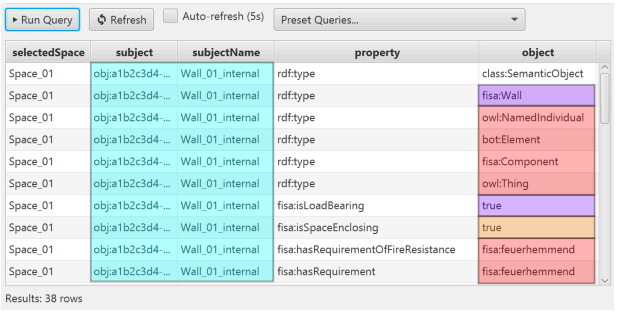

*Figure 6: Results show relationships for `Wall_01`, bounding the selected `Space_01`, in custom Database Viewer (**purple**: user-asserted; **orange**: computed in kernel; **red**: inferred; **lightblue**: `Wall_01`)*

Another trade-off concerns graph management during iteration. Updating existing graphs maintains entity identity but requires synchronization logic, while regenerating graphs sacrifices continuity for simplicity. The framework currently regenerates graphs after each modeling operation, accepting this limitation for implementation tractability. Similarly, the semantic assignment step requires further development. Requiring type assignment during modeling eliminates the generic model problem but may conflict with the exploratory early design, where flexibility takes precedence over fixed commitments. Progressive object type assignment, involving partial assignments that mature with the design, offers a possible resolution but remains unexplored.

Beyond technical constraints, questions remain about how designers will adopt this workflow. Designers who are accustomed to choosing specific types and properties at the end of modeling may resist doing so during the modeling process. Whether handling geometry and semantics at the same time demands too much attention remains untested. The framework assumes that designers benefit from immediate regulatory feedback. However, confirming this would require a carefully developed and constructed user study with designers that compare traditional workflows directly with the proposed approach.

Future work targets full integration with the Partition Platform user interface for parametric operations, which are currently under development. The inference mechanisms demonstrated here for fire safety extend to other regulatory domains where classifications derive from spatial properties and lead to component requirements. Extending the framework to multi-building and urban scales, where site adjacencies and district-level regulations apply, presents additional opportunities for space-driven knowledge approaches.

Conclusion

This paper presents a framework in which modeling operations generate geometric and topological representations and semantic knowledge simultaneously. Partitioning space creates manifold geometries and topological graphs, producing queryable semantic triples, with inferences used to derive classifications and requirements iter-

atively. The proof of concept demonstrated storey classification and the inference of fire safety requirements using the FiSa ontology, thereby establishing the foundations for workflows in which spatial reasoning informs iterative design decisions.

The methodological contribution is a staged workflow that couples topology-aware modeling with inference, applicable beyond fire safety to other regulatory domains where continuous knowledge derivation is preferred over plain validation checking.

References

Beetz, J., Van Leeuwen, J., and De Vries, B. (2009). Ifcowl: A case of transforming express schemas into ontologies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 23(1):89–101.

Donkers, A. and Petrova, E. (2025). Adding geometric functionalities to linked building data using ifcopenshell. In Proceedings of the 2025 European Conference on Computing in Construction, Porto, Portugal.

Eastman, C., Lee, J.-m., Jeong, Y.-s., and Lee, J.-k. (2009). Automatic rule-based checking of building designs. Automation in Construction, 18(8):1011–1033.

Eastman, C. M. (1999). Building Product Models: Computer Environments Supporting Design and Construction. CRC Press, Inc., USA, 1st edition.

Elshani, D., Hernandez, D., Nakhaee, A., Arrascue, A. A., Staab, S., and Wortmann, T. (2026). geof3d: Sparql geometric functions for co-designing buildings. Advanced Engineering Informatics, 71:104261.

Fitkau, I. and Hartmann, T. (2024). An ontology-based approach of automatic compliance checking for structural fire safety requirements. Advanced Engineering Informatics, 59:102314.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2):199–220.

Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). Swrl: A semantic web rule language combining owl and ruleml. Technical report, W3C Member Submission 21 May 2004.

Huhnt, W., Vetter, J. Z., and Sternal, M. (2023). Space partitioning as a holistic alternative to traditional geometric modeling workflows in the aec industry. In Advances in Information Technology in Civil and Building Engineering, pages 415–426. Springer Nature Switzerland.

Jabi, W. and Chatzivasileiadi, A. (2021). Topologic: Exploring spatial reasoning through geometry, topology, and semantics. In Formal Methods in Architecture, pages 277–285. Springer International Publishing.

Massafra, A., Jabi, W., and Gulli, R. (2024). Topological bim for building performance management. Automation in Construction, 166:105628.

Napps, D., Staudt, J., Saluz, U., Chen, X., Steiner, D., Zong, C., Deghim, F., Lang, W., Geyer, P., Schnellenbach-Held, M., Petzold, F., Borrmann, A., and König, M. (2026). Evaluation of building design variants in early phases on the basis of adaptive detailing strategies. Buildings, 16(4):685.

Pauwels, P. and Bod, R. (2014). Architectural design thinking as a form of model-based reasoning. In Model-Based Reasoning in Science and Technology, volume 8, pages 583–608. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pauwels, P., De Meyer, R., and Van Campenhout, J. (2012). Information system support in construction industry with semantic web technologies and/or autonomous reasoning agents. In eWork and eBusiness in Architecture, Engineering and Construction, pages 643–652. CRC Press.

Pauwels, P., van den Bersselaar, E., and Verhelst, L. (2024). Validation of technical requirements for a bim model using semantic web technologies. Advanced Engineering Informatics, 60:102426.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van De Walle, R., and Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. Automation in Construction, 20(5):506–518.

Rasmussen, M. H., Pauwels, P., Hviid, C. A., and Karlshøj, J. (2017). Proposing a central aec ontology that allows for domain specific extensions. In 2017 Lean and Computing in Construction Congress, volume 1, pages 237–244.

Sternal, M. (2025). Polyhedral partition of the euclidean space for consistent spatial modeling of the built environment. PhD thesis, Technische Universität Berlin.

Sternal, M., Oswald, S., and Huhnt, W. (2025). First investigations of advanced operators for consistent spatial modeling the built environment. In ICCCBE 2024, Lecture Notes in Civil Engineering, volume 628, pages 54–67. Springer.

Werbrouck, J., Pauwels, P., Beetz, J., Verborgh, R., and Mannens, E. (2024). Consolid: A federated ecosystem for heterogeneous multi-stakeholder projects. Semantic Web, 15(2):429–460.