

MANAGEMENT OF THE DESIGN KNOWLEDGE CORE IN A CROSS-PLATFORM ENVIRONMENT OF THE BIM PROJECT

Evgenii Ermolenko⁽¹⁾, Bruno Figueiredo⁽²⁾, Miguel Azenha⁽³⁾

(1) University of Minho, EAAD, Lab2PT, ISISE, Guimarães, Portugal, 0009-0003-1228-9550

(2) University of Minho, EAAD, Lab2PT, IN2PAST, Guimarães, Portugal, 0000-0001-8439-7065

(3) University of Minho, ISISE, ARISE, Department of Civil Engineering, Guimarães, Portugal, 0000-0003-1374-9427

Abstract

Effective project data management and consistent communication throughout the project lifecycle are key aspects of BIM. Nevertheless, in current BIM practices, despite rapid advances across the board and in automating specific tasks, design knowledge remains fragmented across files, platforms, and disciplines, limiting its validation and the reuse of design rules. In particular, it becomes a vital issue during iterative design handover among stakeholders. Addressing the limitations above, this paper proposes a framework for management of the shared knowledge core, transposing multimodal architectural knowledge - textual requirements, parametric definitions, and semantic rules - into a unified, object-oriented data model integrated with a BIM environment. By shifting from file-based exchanges to object-level communication, the framework enables automated validation of design intent and the reuse of knowledge datasets. Through a plan-layout case study, the results show how abstract knowledge can be integrated into the BIM lifecycle to enhance coordination and facilitate maturity in the AEC industry.

1. Introduction: Beyond Static IFC Representations

Architectural design is inherently interdisciplinary, emerging from the synthesis of heterogeneous knowledge spanning the spatial, aesthetic, structural, economic, social, physical, and environmental dimensions, as widely recognised in digital design and architectural cognition research [1,2]. While Building Information Modelling (BIM) has advanced to a high degree of automation in project information management, design knowledge remains

fragmented across various domain-specific datasets that are continuously reconstructed across tools and project phases, resulting in redundancy, loss of original intent, and limited traceability (Figure 1, "As-Is"). The Industry Foundation Classes (IFC) schema (ISO 16739) provides a robust model for physical component exchange, but offers limited support for the parametric relationships and abstract constraints that drive design logic [3,4].

Recent research on rule-based and knowledge-aware BIM has converged on three strands: (1) IFC enrichment with parametric and topological semantics [11], (2) graph-based interoperability layers maintaining cross-domain consistency [12], and (3) ontology-driven rule checking using SWRL or SHACL on top of OWL representations of building elements. Initiatives such as buildingSMART's IDS (Information Delivery Specification) formalise the information that must be exchanged, but they remain declarative and stop at the property level, leaving the encoding of design intent and parametric relationships to ad hoc workflows [4,5]. The framework proposed here complements these directions by maintaining a project-scoped, object-level knowledge core that links abstract grammars to BIM instances at runtime, rather than baking constraints into a static schema.

The predominance of file-based workflows further exacerbates this issue. Currently, architectural design largely adheres to handover principles throughout the development process. Consequently, with each transfer of design outputs among stakeholders, the design rules established before decisions are lost. Each transition causes a partial loss of intent because only final states are exchanged as detached files, not the underlying rules and constraints that defined them [5]. Existing integrations between parametric environments (e.g., Grasshopper/Dynamo/Rhino/Inside.Revit) and BIM platforms remain constrained by platform-specific definitions, resulting in the replication of abstract logic rather than its sharing [4]. To address the fragmentation of architectural knowledge inherent in conventional design workflows, a shift is required from detailed representations to abstract, object-level knowledge, thereby establishing a common semantic tier that connects domain-specific parts of the model without duplicating information [2]. In such a holistic approach, parts share high-level semantic properties and behaviour of the core grammars, while focusing on low-level, detailed domain representation - geometric, analytical, or visual. Accordingly, a knowledge management system is needed to manage grammatical transitions and validation between domains and the project's knowledge core. In response, this paper proposes a framework for building such a system that will evolve throughout the project lifecycle, assisting the development process. The framework allows formalisation of design intent as semantic rules linked to object definitions, enabling continuous validation, reasoning, and reuse across platforms and disciplines. The system architecture described in the subsequent sections is designed to support cross-platform and object-level communication, AI-assisted inference on project data, and explainable validation integrated with the Open BIM environment [6].

2. System Framework

The proposed Shared Knowledge Core replaces file-based exchange with object-level semantic communication [7]. The system adopts a Core-Periphery data model, in which abstract design objects and rules form a shared semantic core that connects heterogeneous domain representations [8]. The system is structured into five layers, organised to support continuous knowledge ingestion, reasoning, and validation across the project lifecycle (Figure 1, "To-Be").

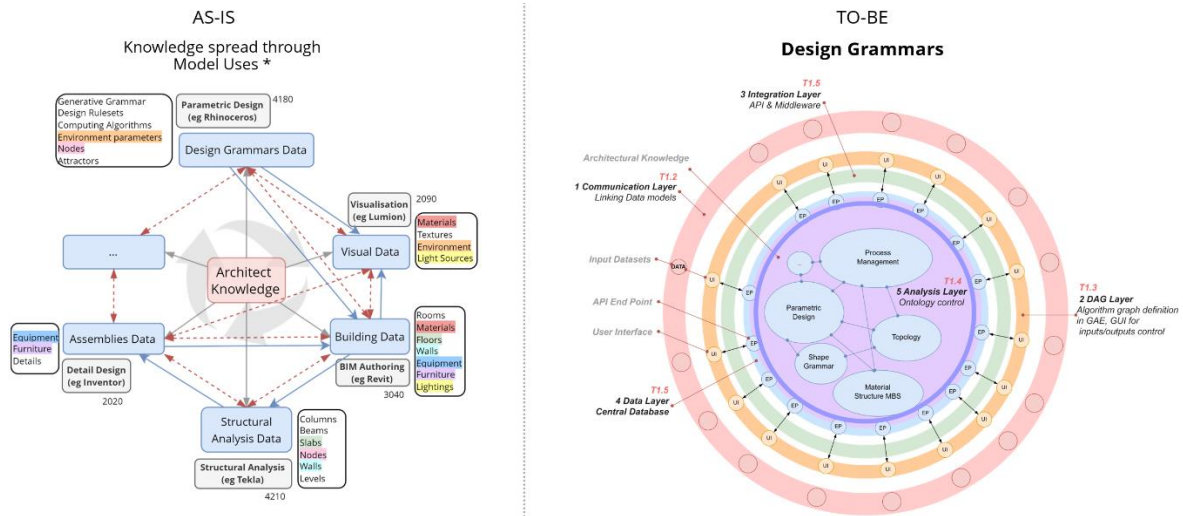


Figure 1: System framework (As-Is vs To-Be)

2.1 Architecture Foundation: Domains of interaction

The architecture comprises two domains (Figure 2). The Model Domain holds geometric and domain-specific semantic representations, such as BIM models. The Grammar Domain encapsulates design intent through formal rules, constraints, and semantic relationships. The two connect at a common graph-based knowledge layer, in which Model Domain objects are linked to grammatical structures that determine their permitted configurations and behaviour. Interaction between domains is orchestrated through a workflow automation environment in the n8n platform. A locally hosted LLM (llama3.1 via Ollama, low temperature for deterministic output) assists with rule interpretation and query generation; schema constraints and few-shot examples are embedded in the prompt at inference time rather than baked in during fine-tuning, keeping the system schema-agnostic and updatable. All reasoning, however, remains grounded in explicit grammars and ontological definitions: the LLM produces candidate Cypher and SWRL atoms that are post-validated against the active schema before any commitment to the graph. These domains underpin the layered architecture described next, beginning with the Communication Layer.

2.2 Communication Layer: Mediating Grammars

Users interact with the system through an interface that accepts heterogeneous inputs, including natural language commands, parametric definitions, and BIM-derived datasets (Figure 2, "1 Layer"). In the Grammars Domain, users provide design rules, including spatial programs, regulatory constraints, or design guidelines, which are then interpreted by a grammar-intelligent parser and transmitted via the Model Context Protocol (MCP) to the Core (Figure 1, "1G") To facilitate formal reasoning and validation, grammars are encoded using the Semantic Web Rule Language (SWRL), which decomposes design intent from the user inputs in Natural Language (NL) into atomic rules defined over ontological classes and properties [9]. For instance, the constraint "the maximum building height is 75 m" is encoded as a single form SWRL rule: "Building(?b)^hasHeightM(?b,?h)^swrlb:greaterThan(?h,75)→violatesMaxHeight(?b, true)." Consequently, the framework implies a two-level grammar management: users create and refine project-specific rules that reflect design intent and constraints, while the ontology ensures

structural consistency of the encoded inputs and prevents conflicts. After passing through such a controlled yet adaptable parsing mechanism, the input data is posted in a dedicated graph database, which further serves as the central repository for the models' reasoning, validation, and inference (Figure 2, "2 Layer"). Within the Model Domain, objects derived from BIM models are mapped to semantic entities and associated with their corresponding classes and properties in the graph database (Figure 2, "1M"). A dedicated semantic reasoning pipeline establishes these associations and requests relevant constraints from the knowledge core (Figure 2, "5 Layer"). Constraint retrieval and contextualisation are handled through a retrieval-based reasoning mechanism that queries the graph and returns project-specific rule sets and constraints applicable to the current design state. These constraints are then used to validate, evaluate, or populate the model solution space in response to evolving design conditions. Live connectors between the visual programming environment and BIM platforms maintain shared object references and parameter states across domains. A parametric definition, built using Directed Acyclic Graph (DAG) technology, is one of the key components in this setup, encoding dependencies, constraints, and transformation rules rather than fixed geometric configurations [1]. In this framework, parametric models are treated as mutable carriers of design intent, capable of synchronising multiple downstream representations, including BIM models and analytical simulations. Consequently, the Communication Layer enables bidirectional data exchange, in which semantic constraints from the Shared Knowledge Core guide parametric behaviour of model objects while ongoing BIM model updates feed back into system reasoning. As a result, design states become provisional and can be evaluated throughout iterative development.

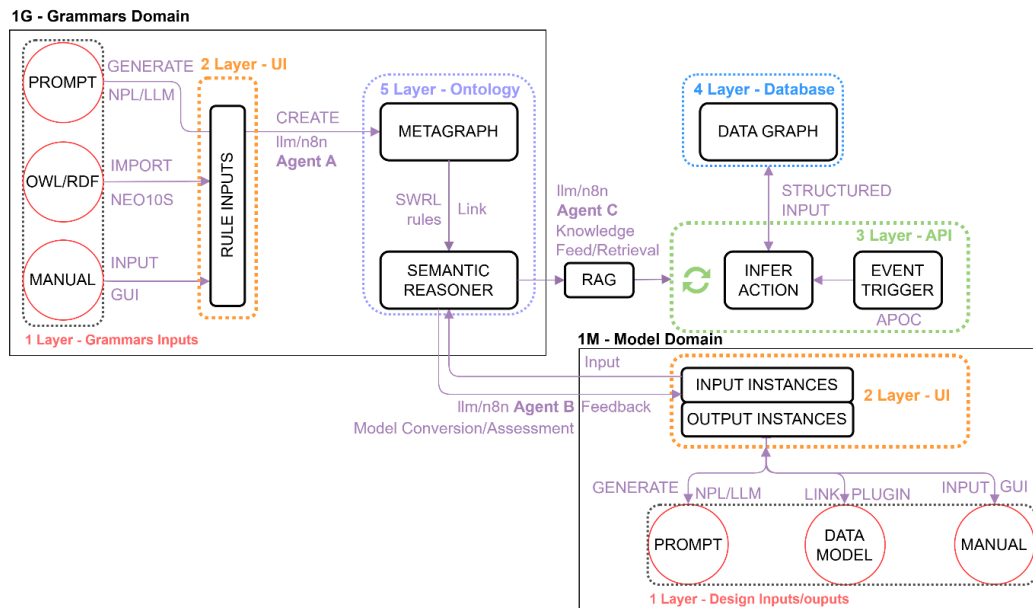


Figure 2: Design Knowledge System Domains

2.3 Data Layer: Single-Source Object Integrity

While the Communication Layer provides interfaces and mechanisms for translating heterogeneous inputs into structured abstract grammars, the Data Layer forms the persistent

storage of shared architectural knowledge. The primary challenge addressed at this level involves the integration of heterogeneous architectural data - including geometric, semantic, temporal, and unstructured information - within a unified object-oriented framework, while preventing duplication or fragmentation of data across different domains. To address this challenge, the Data Layer is implemented as a modular extract–transform–load (ETL) pipeline that enables controlled ingestion, transformation, and synchronisation of data originating from diverse design environments [10]. Raw inputs received via the Communication Layer are parsed and aligned with a predefined ontological schema, ensuring semantic coherence and consistent object definitions. Once transformed, data is stored in specialised, yet interoperable, storage domains, each optimised for a specific data type and access pattern.

Rather than enforcing a single database paradigm, the system adopts a multi-domain data architecture in which architectural knowledge is distributed across complementary storage domains connected through shared identifiers and controlled interfaces. These domains include: a relational domain for structured BIM data aligned with the IFC schema [3]; a spatial domain for geometric and locational information; a graph domain for semantic data, ontologies, and design grammars [5,11]; a time-series domain for simulation results and performance metrics; and a document-oriented domain for unstructured content such as textual annotations, sketches, and reusable knowledge assets. Although conceptually distinct, these domains operate as a coherent whole through the ETL pipeline and API-mediated access.

In the current implementation, the graph domain functions as the central knowledge repository, storing ontological structures and formalised design grammars. A graph-based representation is particularly suited to design knowledge, which is characterised by evolving schemas, complex relationships, and constraint-based dependencies. Semantic hierarchies, topological relations, and rule structures can be expressed and queried more naturally in a graph model than in rigid relational schemas, supporting scalable reasoning and transparent validation [12].

A core architectural principle of the Data Layer is single-source object integrity. Each abstract object is explicitly defined within a single domain and referenced by other objects as needed, thereby preventing redundancy and inconsistency. For example, material definitions, spatial classifications, or rule entities are not replicated across BIM, analysis, or visualisation environments; instead, they are queried dynamically from the Shared Knowledge Core. This principle directly addresses common shortcomings of file-based workflows, where duplicated definitions lead to misalignment and loss of intent.

Access to the Data Layer is mediated through the Application Programming Interface (API) and the Model Context Protocol (MCP), enabling scalability for interaction with potential external tools, AI-assisted reasoning pipelines, and user interfaces. By separating storage concerns while maintaining semantic cohesion at the object level, the Data Layer provides a robust foundation for continuous validation, explainable inference, and knowledge reuse across design stages and platforms.

Meanwhile, the Data Layer is designed to coexist with rather than replace IFC. Object identifiers in the graph reference IFC GUIDs where applicable, and the relational subdomain mirrors the IFC class hierarchy (ISO 16739), ensuring exchanges with conventional BIM tooling remain valid [3]. With respect to buildingSMART's Information Delivery Specification (IDS), the framework treats IDS as a complementary declarative layer: an IDS document specifies which properties must be present for a given exchange. In contrast, the Knowledge Core specifies why those properties are constrained and how the constraint should behave under

iterative design changes. The two layers can be aligned by mapping IDS specifications to SWRL rules at ingestion time, an integration left for future work.

2.4 Analysis Layer: Consecutive Validation

The Analysis Layer executes reasoning, validation, and inference processes over the architectural knowledge stored in the Data Layer. Rather than performing static, post-hoc checks, this layer enables continuous evaluation of evolving design states by operating on abstract objects and formalised design grammars. Analytical processes are organised as modular pipelines that can be triggered by user queries, model updates, or rule modifications originating from the Communication Layer.

Analytical pipelines are orchestrated within a workflow automation environment (n8n). Each pipeline is selected and configured automatically based on the type of input data and the intended analytical task. Before analysis is performed, all model inputs are semantically aligned with the system ontology and associated with corresponding grammar entities, ensuring that reasoning operates on structured, project-specific knowledge rather than on uncontextualised language model output, avoiding the "black box" nature of typical AI tools.

Following semantic alignment, the Analysis Layer generates structured database queries to retrieve relevant objects, relationships, and constraints from the Data Layer. Retrieved data is then processed to perform rule-based validation, constraint satisfaction checks, or topological reasoning against the current design model state. The results of these evaluations are translated into meaningful feedback that guides further design exploration.

By integrating formal reasoning, semantic constraints, and AI-assisted mediation, the Analysis Layer enables scalable, explainable validation of design intent across platforms and disciplines.

2.5 Operator effort and learning curve

Two operator profiles interact with the framework. Domain experts (typically planners or lead architects) author rules in natural language and curate the ontology. Computational designers connect parametric and BIM environments to the Knowledge Core through the Grasshopper plugin. The framework deliberately offloads SWRL and Cypher construction to a constrained LLM pipeline at inference time [9], so that no end user needs to write SWRL by hand. Schema constraints, atom-naming conventions, and a few-shot example are embedded in the prompt rather than encoded through model fine-tuning, which keeps the system updatable as the ontology evolves.

In informal sessions during the prototype's development, a domain expert with no prior SWRL background authored a first valid rule (a height-limit constraint) in approximately 15 minutes, including iterative refinement of the natural-language phrasing. Onboarding a computational designer to the Grasshopper components required roughly two hours of guided work, after which the operator could load rules, bind Rhino geometry, evaluate constraints, and publish a Speckle overlay independently. Configuration of the underlying ontological scaffolding — class hierarchies, datatype properties, and IRI conventions — remains a one-off task best handled by an architect familiar with formal vocabularies and is estimated at one working day for a project of medium scope.

3. System Implementation

The proposed Design Grammars system was evaluated through three typical design tasks: (1) a plan layout generation task, (2) iterative validation of evolving design states against rules, and (3) ingestion and retrieval of project documentation through the Knowledge Core, assessing its semantic and generative capabilities. Throughout the workflow, the architect remains the primary decision-maker, using the system to explore, validate, store and retrieve design data rather than to automate design outcomes. Further, we will consider the plan layout generation task in detail.

3.1 Plan-layout case study

The case study evaluated the framework on a residential plan-layout task (1) with two coupled objectives: encoding a spatial program from an existing BIM model, and reusing it to generate alternatives. Spaces and adjacency relationships were parsed as semantic entities and stored in the graph-based knowledge core, abstracting the original design from its geometric configuration while preserving spatial hierarchy, connectivity, and functional intent (Figure 3). In a second stage, the encoded program and user-defined adjacency, circulation, and spatial constraints were retrieved through a live connector and applied to a parametric generator in Grasshopper with Kangaroo Physics [13] (Figure 4). Rather than yielding a single deterministic solution, the Knowledge Core enabled exploration of a solution space of valid configurations, dynamically evaluated against the encoded grammars as parameters changed.

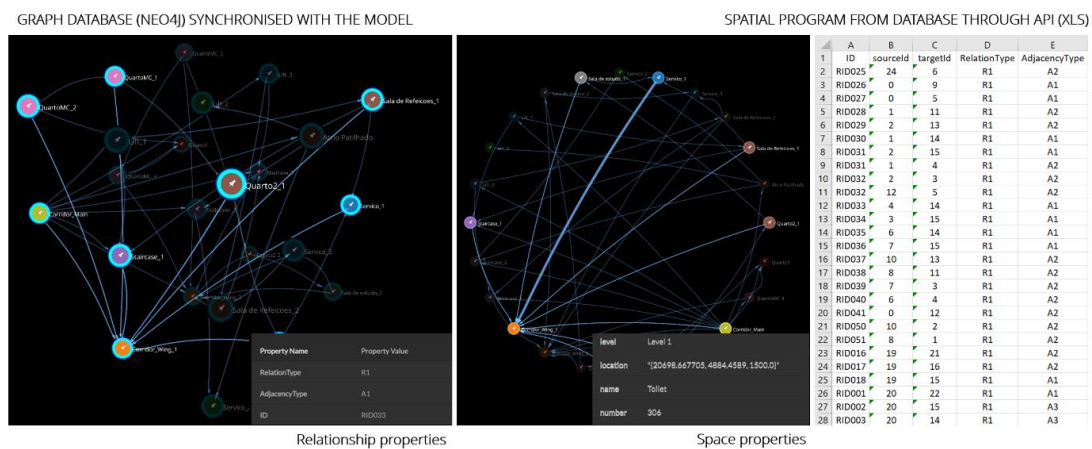


Figure 3: First Case study subtask. Spatial Program encoded in Design Grammars

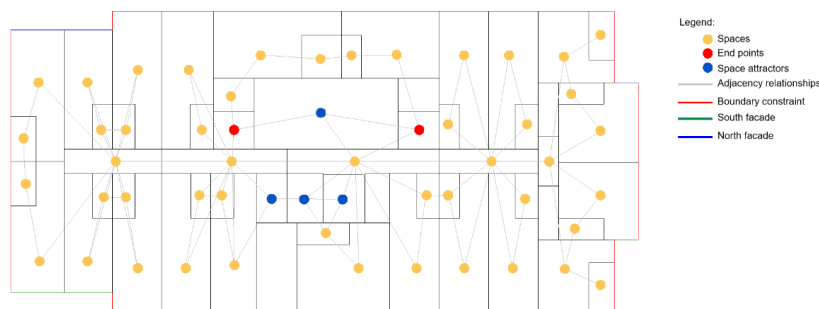


Figure 4: Second Case study subtask. Generated layout result

3.2 Design-state validation and project documentation

Two other case studies tested (2) iterative validation of evolving design states and (3) ingestion and retrieval of project documentation through the Knowledge Core. (2) In an urban-block study, a lead architect ingested eight regulatory rules in natural language via the web platform — covering height, site coverage, inter-building spacing, minimum floor area, direct-sunlight hours, and adjacency — while a computational designer bound them to a Grasshopper model through the Design Grammars connector. A Design State component serialises typed parameter inputs (numeric, integer, and boolean) alongside each validation run, allowing past states to be reinstated and alternatives compared without losing constraint context. Results are published to the platform, yielding a per-run history queryable by rule and by state. (3) In the last case study, unstructured specifications and notes were ingested into a dedicated Knowledge Core partition, decomposed into note and tag nodes with full-text indexing. An LLM-mediated workflow performs hybrid search-then-summarisation on natural-language queries and supports operator-mediated note edits with explicit commit. Ingest, query, and update interactions are persisted as session nodes, keeping each artefact's design history traceable.

3.3 Indicative metrics

Table 1 reports indicative metrics from the prototype evaluation across the three case studies, comparing a conventional file-based approach with the Knowledge Core prototype on rule authoring effort, propagation of constraint changes, breadth of design exploration, intent retention across handover, and core system performance.

Table 1: Indicative metrics from the prototype evaluation across case studies.

Indicator	Conventional approach	Knowledge Core prototype	Notes
Time to author and validate a single rule	~45 min (manual or scripted check)	~15 min (NL prompt + model validation)	First-time user, height-limit rule
Time to propagate a constraint change across alternatives	~30 min per alternative (manual edits)	<2 min (single edit, automatic re-evaluation)	Three-alternative comparison
Iterations explored per design session	3–5 alternatives	12–20 alternatives	Plan-layout case study
Coverage of original design intent is retained after a single handover	Estimated 60–70% (parametric definition + rules)	~95% (constraints retained as graph)	Self-reported; intent expressed as rules

Current implementation has shown limitations inferred from case studies. SWRL coverage is restricted to ClassAtom, DataPropertyAtom, and BuiltinAtom patterns; full OWL/SWRL expressivity is not handled. LLM-generated Cypher is heuristically post-validated, but occasional malformed outputs require human review. Project isolation relies on a property filter within a single Neo4j instance, which is acceptable for a pilot but should be revisited for multi-tenant deployment. Finally, the evaluation rests on controlled academic case studies, and graph query scalability has not yet been stress-tested at production scale.

4. Conclusion

The proposed Shared Knowledge Core framework demonstrates a feasible path for encoding, managing, and reusing grammar knowledge throughout the design lifecycle. By integrating multimodal design rules, parametric definitions, and BIM models into a unified semantic environment, it addresses, at the level of a working prototype, several persistent challenges in architectural design communication, validation, and coordination. Its modular data architecture and AI-assisted workflows support dynamic reasoning across evolving design states and link formal requirements to specific model representations in a transparent, explainable manner. The case studies presented here show the feasibility of the approach across plan-layout generation, iterative design-state validation, and structured ingestion of project documentation. The reported metrics are preliminary and were obtained in controlled academic conditions; broader empirical validation in production AEC settings, scalability stress testing, and integration with declarative exchange standards such as IDS remain priorities for future work. Within these caveats, the framework offers a flexible foundation for scalable knowledge representation and intelligent design reasoning aligned with the Open BIM agenda.

Acknowledgements

This work is financed by national funds through FCT – Foundation for Science and Technology, under grant agreement 2024.03763.BD attributed to the 1st author, and by the "R2U Technologies | modular systems" project (ref.: C644876810-00000019), which is funded by PRR – Plano de Recuperação e Resiliência – and by the European Funds Next Generation EU, under the incentive system "Agendas para a Inovação Empresarial". The initiative is supported through the Multiannual Funding of the Landscape, Heritage and Territory Laboratory (Lab2PT), Ref. UID/04509: Laboratório de Paisagens, Património e Território (Lab2PT/UM), financed by national funds (PIDDAC) through the FCT. This work was also supported by FCT under the R&D Unit Institute for Sustainability and Innovation in Structural Engineering (ISISE), under the references UID/4029/2025 (<https://doi.org/10.54499/UID/04029/2025>) and UID/PRR/04029/2025 (<https://doi.org/10.54499/UID/PRR/04029/2025>), and under the Associate Laboratory Advanced Production and Intelligent Systems ARISE under reference LA/P/0112/2020.

References

- [1] R. Oxman and R. Oxman, *Theories of the Digital in Architecture*. Abingdon: Routledge, Taylor and Francis, 2014.
- [2] B. Lawson, 'How Designers Think – The Design Process Demystified', *University Press, Cambridge*, Jan. 2006.
- [3] 'ISO 16739-1:2024 - Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries — Part 1: Data schema'.
- [4] Á. Semjén and J. Szép, 'Integrating generative and parametric design with BIM: A literature review of challenges and research gaps in construction design', *Applications in*

- Engineering Science*, vol. 23, p. 100253, Sep. 2025, doi: 10.1016/J.APPLS.2025.100253.
- [5] N. Alassaf, 'Computational Precedent-Based Instruction (CPBI): Integrating Precedents and BIM-Based Parametric Modeling in Architectural Design Studio', *Buildings*, vol. 15, no. 8, Apr. 2025, doi: 10.3390/BUILDINGS15081287.
- [6] R. Sacks, M. Bloch, M. Katz, R. Yosef, 'Knowledge-based OpenBIM data exchange for building design', *Autom. Constr.*, vol. 156, p. 105144, 2023.
- [7] S. Kyratzi and P. Azariadis, 'Integrated Design Intent of 3D Parametric Models', *CAD Computer Aided Design*, vol. 146, May 2022, doi: 10.1016/j.cad.2022.103198.
- [8] M. Papachristou and J. Kleinberg, 'Core-periphery Models for Hypergraphs', *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1337–1347, Aug. 2022, doi: 10.1145/3534678.3539272.
- [9] S. Abrishami, J. Goulding, F. Pour Rahimian, and A. Ganah, 'Virtual generative BIM workspace for maximising AEC conceptual design innovation: A paradigm of future opportunities', *Construction Innovation*, vol. 15, no. 1, pp. 24–41, Jan. 2015, doi: 10.1108/CI-07-2014-0036.
- [10] J. L. Coenders, 'Next generation parametric design', *Journal of the International Association for Shell and Spatial Structures*, vol. 62, no. 2, pp. 153–166, Jun. 2021, doi: 10.20898/J.IASS.2021.018.
- [11] J. Wu, S. Esser, S. Nousias, and A. Borrmann, 'Enriching IFC Models with Spatial Design Logic and Parametrics to Improve Design Adaptability – The Case of Alignment Grids', *Lecture Notes in Civil Engineering*, vol. 628 LNCE, pp. 91–105, 2025, doi: 10.1007/978-3-031-84208-5_8.
- [12] C. Preidel, A. Borrmann, H. Mattern, M. König, S.-E. Schapke, 'Graph-based inter-domain consistency maintenance for BIM models', *Autom. Constr.*, vol. 154, p. 104979, 2023.
- [13] 'Kangaroo Physics | Food4Rhino'. Accessed: Apr. 16, 2025. [Online]. Available: <https://www.food4rhino.com/en/app/kangaroo-physics>