A Model-driven BIM2SIM Workflow for Surrogate Modeling

Yi Wang
yi.wang@uibk.ac.at
University of Innsbruck
Innsbruck, Austria

Claudio Gomes
claudio.gomes@ece.au.dk
Aarhus University
Aarhus, Denmark

Philipp Zech
philipp.zech@uibk.ac.at
University of Innsbruck
Innsbruck, Austria

Abstract

Digital twins enable monitoring, control, optimization, and predictive maintenance in buildings, but they require fast simulations to operate in (near) real-time during building operation. Surrogate models address this computational challenge, yet training them demands large datasets. Current approaches face a fundamental bottleneck: operational data only becomes available after commissioning, while existing BIM2SIM workflows remain manual, error-prone, and cannot scale to generate the necessary amount of synthetic training data. To this end, we present a model-driven pipeline that automates the BIM2SIM workflow. Our method transforms Building Information Models (BIM) into executable simulation models (SIM) that generate synthetic training data at scale for surrogate modeling. We validate our approach using a Living Lab at the University of Innsbruck. Results show that our generated simulation models match measured building behavior, the code generation scales with model complexity, and the synthetic data is sufficient for training reliable surrogate models.

CCS Concepts

• **Software and its engineering** → **Domain specific languages**; • **Computing methodologies** → **Modeling methodologies**; **Simulation support systems**; *Machine learning*; • **Applied computing** → *Computer-aided design.*

Keywords

MDE, MBSE, Building Information Modeling, Surrogate Modeling

ACM Reference Format:
Yi Wang, Claudio Gomes, and Philipp Zech. 2026. A Model-driven BIM2SIM Workflow for Surrogate Modeling. In *Proceedings of ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '26).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 Introduction

Digital twins (DTs) have emerged as transformative tools that create virtual representations of physical assets, enabling real-time monitoring, analysis, and optimization along their lifecycle [56]. The Architecture, Engineering, Construction, and Operations (AECO) industry has witnessed significant growth in DT adoption, driven by DTs' demonstrated capacity to decrease operational energy consumption and carbon emissions in buildings [40]. This is achieved through continuous performance monitoring, fault detection, predictive control and maintenance strategies, design scenario exploration, virtual commissioning processes, and data-driven decision-making [22]. To achieve these benefits, DTs in AECO integrate several key technologies into a unified platform: Building Information Modeling (BIM), the Internet of Things (IoT), cloud and edge computing, simulation, and artificial intelligence [57].

At runtime (i.e., after commissioning), the building control system (BCS) and the DT maintain bidirectional data exchange: the BCS provides operational data to the DT, which returns optimized control commands. By fusing multi-sensor data with model predictions, the DT analyzes current conditions and predicts future states while providing actionable insights for decision support and optimization [55]. Central to this capability is simulation, which enables DTs to assess the energy-performance gap (the discrepancy between predicted and actual energy use [39, 61]) to update control strategies, or supports fault detection. Because design-phase simulations rely heavily on assumptions, actual occupant patterns and system dynamics only emerge during operation, making reliable operational simulations essential for these DT applications.

However, generating simulation models from building design artifacts remains a significant bottleneck. A building's design model typically is implemented in a dedicated BIM tool. Building Information Modeling (BIM) describes a model-driven process that employs object-based modeling to generate semantically rich, *n*-dimensional representations of a building's physical and functional characteristics, encompassing geometry, cost, scheduling, performance, and support for collaborative design, clash detection, and full lifecycle management [5]. However, BIM for now lacks explicit support for modeling a BCS, dynamic system behavior, and as-built variances [28, 29, 49]. Although emerging research has explored real-time data, IoT connectivity, and operational processes [60, 62], mature and standardized approaches for modeling such behavioral and control semantics have yet to be established.

Furthermore, despite the prevalent adoption of semi-automated BIM2SIM workflows, several key gaps persist [14, 34, 63]: heavy reliance on manual geometric remediation to fix fragmented space boundaries; semantic data loss during model exchange that necessitates redundant attribute re-entry [28]; lack of automated regression mechanisms to synchronize simulations with continuous design changes [30]; and insufficient representation of dynamic operational aspects (e.g., HVAC systems) within static BIM models [63]. These issues require heavy manual intervention [7], and the resulting *media breaks* at the model level degrade the fidelity of downstream simulation workflows. These limitations become particularly critical when moving to large and complex buildings and their BCS. In such settings, the number of assets, interfaces,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MODELS '26, Malaga, Spain
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/26/XX
https://doi.org/10.1145/XXXXXXX.XXXXXXX

and control relationships increases substantially, making manual intervention and media breaks a major bottleneck for producing and maintaining simulation-ready artifacts.

The impact of these BIM2SIM limitations extends to multiple DT applications. For iterative design analysis and virtual commissioning, manual workflows prevent rapid exploration of control strategies and design variants [63]. For operational optimization, the inability to efficiently update simulation models as building configurations change hinders continuous performance assessment [62]. For emerging applications such as surrogate model training, where machine learning models require extensive simulation-derived datasets to serve as fast runtime predictors within DTs [13, 59], the current BIM2SIM bottlenecks prevent generating the diverse, large-scale datasets needed for accurate training. This creates a clear need for efficient and highly automated BIM2SIM workflows. As discussed in Section 2, existing approaches still do not provide effective solutions.

Model-driven engineering (MDE) provides a robust framework to address the aforesaid media breaks and the resulting semantic gaps [51]. Specifically, MDE allows for the engineering of language extensions for explicit modeling of simulation-relevant semantics, and the generation of well-formed executable models that are suitable for high-fidelity (e.g., minute-level) operational simulation. Leveraging this potential, we combine BIM with MDE, specifically Model-based Systems Engineering (MBSE) [8, 47], to capture system-level requirements, control logic, interface definitions, and operational parameters in BIM models for subsequent simulation model generation [24]. This leads directly to the research questions (RQs) of our work:

RQ1: What semantic structures and levels of abstraction are necessary within MBSE-aligned BIM to formally capture the system view of buildings concerning their behavioral semantics and operational intent? Design models typically omit explicit representations of BCS and operational aspects, and limited semantic interoperability prevents dynamic and control semantics from being translated into simulation-ready models. We address two technical dimensions: (i) modeling semantics that unambiguously capture behavior and control aspects, and (ii) representation of control-to-physical mappings and as-built data to enable requirements tracing and verification.

RQ2: How can MBSE-aligned BIM artifacts be operationalized to automatically and efficiently generate executable simulation artifacts? Automated translation from MBSE-aligned BIM to simulation must preserve semantic fidelity, and systematic simulation management requires robust structural design for scenario encapsulation and co-simulation workflows. We focus on two technical dimensions: (i) deterministic model-to-text transformations that translate MBSE-aligned BIM models into executable simulation components, and (ii) formalized execution context definitions to encapsulate solver configurations and boundary conditions for consistency across coupled domains.

Commensurate with our **RQs**, we introduce a model-driven approach to streamline BIM2SIM workflows and their underlying automation pipelines. Motivated by MDE's capacity to address the challenges above, we integrate BIM with MBSE to enrich static design models with operational runtime aspects and automate the generation of executable simulation artifacts. Specifically, we develop an MBSE-aligned Domain-Specific Modeling Language (DSML) [20] that extends static BIM artifacts with system behavioral semantics, a DSML to capture operational semantics, and an automated model translation that transforms these enriched models into fully executable simulation artifacts. We evaluate our approach through a Living Lab case study, validating simulation accuracy against measured data, assessing the efficiency and scalability of the generation pipeline, and demonstrating applicability to surrogate model training scenarios.

Section 2 reviews related work and identifies the research gap. Section 3 presents our solution proposal followed by Section 4 which details its implementation. Section 5 evaluates the approach through a Living Lab case study. Section 6 discusses our results before we conclude in Section 7.

2 Related Work

MBSE formalizes system requirements, functions, structures, and behaviors into coherent models, primarily aimed at enhancing traceability, reusability, and consistency across the development lifecycle [47] and has been applied to simulation to address the silos between systems engineering and domain simulation. Banagar et al. [6] proposed a toolchain that leverages SysML V2 to manage simulation configurations. Their work introduced the *Standard System Interface* workflow, which enables automated transformation of SysML V2 architectural models into executable models compliant with the `System Structure and Parameterization` standard. Gu et al. [26] proposed X-SEM, a modeling and simulation-based systems engineering methodology, to bridge the gap between system modeling and simulation at the methodological level. X-SEM fuses system architecture definitions with physical behavior descriptions through a unified modeling and simulation language (X Language). However, the reliance on a specialized language limits its interoperability with the standard SysML toolchains.

In the context of BIM2SIM, Alhammad et al. [2] conducted a systematic review of this domain, categorizing existing integration approaches into Industry Foundation Classes (IFC)-based and Green Building XML (gbXML)-based workflows. Their analysis reveals that, despite tool support, model exchange remains a critical bottleneck. In particular, the transformation from design to simulation models often suffers from information loss and misinterpretation. Gabriela et al. [7] reinforce these issues within their semi-automated BIM2SIM framework that requires repetitive manual configuration and exhibits critical geometric errors like model shrinkage and missing thermal zones. To address such interoperability challenges, Zech et al. [64] introduced a rule-based model transformation to automate model translation between standards like IFC and gbXML. Related to this, Andriamamonjy et al. [3] and Remmen et al. [48] successfully automate the transformation of IFC models into Modelica for building energy performance simulation. Kong et al. [38] integrated FMI with ontologies to assemble modular, multi-domain co-simulation networks from BIM models. However, while these pipelines significantly advance the generation of simulation models with physical logic, they typically lack modeling of complete BCS installations. Consequently, these works struggle to natively integrate cross-domain, complex dynamic control logic which is required for high-fidelity operational simulations. These

issues are further compounded by lacking scalability, as highlighted by Souza et al. [53] who demonstrate that for smart urban contexts, the integration between BIM platforms and environmental simulation tools is hampered by interoperability limitations and a lack of standardization.

More recent work has targeted the integration of system-level management into BIM workflows. For instance, Sandberg et al. [50] introduced a BIM-based master model to capture design variables for optimizing energy use and cost. Gourlis et al. [23] employed continuous and discrete simulations of industrial production and building thermal performance. However, these approaches lack rigorous modeling infrastructures, which renders automated semantic translation difficult. Contrary to these works, Bolshakov et al. [11] proposed a conceptual framework integrating BIM and MBSE to optimize facility management by formalizing non-physical entities such as requirements, functions, and processes in BIM models to enable simulation-based optimization of resource consumption and operational workflows. However, their work remains primarily at a conceptual level. More practically, Murphy et al. [42] proposed a framework that maps BIM to SysML-compatible metamodels. This integration lays the conceptual foundation for integrating dynamic simulation and real-time regulatory validation within DTs. In reality, Murphy et al. automate static compliance checking instead of dynamic performance simulation and their approach r requires manual tuning for translating hierarchical elements.

In summary, existing methods remain fragmented or functionally limited. MBSE offers rigorous behavior modeling but lacks the physical and spatial context of buildings. BIM2SIM workflows, in contrast, struggle with information loss, manual reconfiguration, and lack of capturing control logic. Recent efforts fall short of a comprehensive executable solution: non-standardized system-level architectures lack rigorous semantics, and formal MBSE-BIM integrations remain conceptual or lack full automation [9, 27]. This highlights a critical gap: the lack of an automated model-driven pipeline capable of integrating both spatial-physical constraints and system-level behavioral semantics for dynamic operational simulations. To this end, we propose a systematic MBSE-aligned BIM approach that employs DSMLs to augment standard BIM representations with control and operational semantics for automated synthesis of simulation-ready models.

3 Solution Proposal

Commensurate with the identified issues of semantic gaps and media breaks, we propose a model-driven approach to establish an automated BIM2SIM workflow. By leveraging MDE's capabilities in metamodeling and model transformations, our approach avoids information loss and manual inconsistencies that degrade simulation fidelity. This ensures consistent generation of executable simulation models, which also facilitates the acquisition of high-fidelity synthetic data for surrogate model training. Fig. 1 illustrates the workflow of our proposed solution.

Our model-driven pipeline is integrated into Revit [4], a widely used state-of-the-art BIM tool [54]. To address the lack of native simulation semantics in BIM, we introduce two DSMLs that extend Revit's internal modeling facilities, enabling practitioners to explicitly model and configure simulation parameters and control

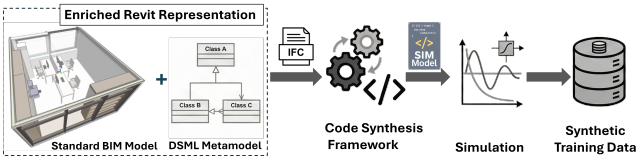


Figure 1: Our model-driven pipeline transforms enriched BIM models into simulation-ready models using model-to-model transformations. Simulation models are then used for design assessment and, in the course of this, generating data for surrogate models training.

logic directly alongside building design. Automated model transformations translate these enriched models into executable Modelica models. We selected Modelica [41] as our simulation platform for its open-source nature and robust multi-physics coupling support, which provides significant advantages over single-domain tools. The execution of these simulations generates large amounts of high-fidelity synthetic data for training surrogate models. We claim the following contributions

A model-driven pipeline that eliminates media breaks between BIM and simulation
Two DSMLs that extend BIM with explicit system and simulation semantics and control logic
Automated model transformations from enriched BIM models to executable Modelica models
Demonstrated feasibility and practicality through Living Lab validation and surrogate model training

Our research follows Design Science Research (DSR) [31] and tackles a persistent challenge: *How to bridge semantic gaps and media breaks in BIM2SIM workflows to enable automated generation of simulation artifacts for large-scale synthetic data production?* We present an integrated model-driven pipeline (the *artifact*) that addresses this problem and demonstrate its feasibility and practicality in Section 5.

4 Solution Artifact

We now detail our model-driven approach, which uses metamodels, a modeling methodology, and a code synthesis framework to bridge the semantic gap between BIM and dynamic simulations.

4.1 Metamodels

The semantic foundation of our approach capitalizes on two DSMLs that capture the necessary semantics for high-fidelity simulation. These are structured as two complementary metamodels: the Domain Metamodel (DM), which serves as a semantic enrichment layer over standard BIM models, and the Control Logic Metamodel (CLM), which models executable controller behavior for simulation dynamics. To epitomize these concepts, we define a small running example (cf. Fig. 2): a single-zone example that instantiates the key DM and CLM constructs in a compact setting. It includes both an HVAC temperature control loop (room temperature sensor, VAV terminal box, PID controller) and a lighting control loop (illuminance sensor, dimmable luminaire, lighting controller).

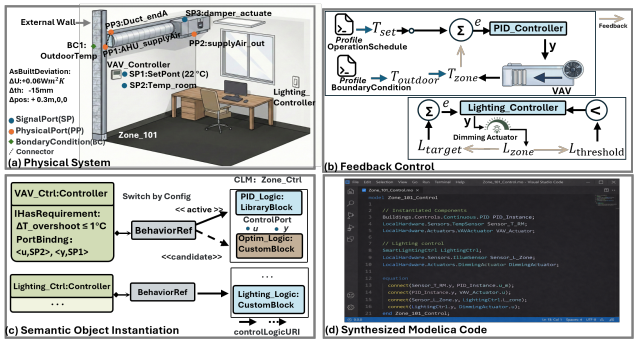


Figure 2: Running example: a single-zone office with HVAC and lighting control epitomizing key DM and CLM concepts.

4.1.1 DM: Semantic Enrichment Representation. The DM (cf. Fig. 3) elevates standard BIM models from simple design models to specifications that contain all information necessary to generate simulation models by embedding the controller-interface semantics, operational intent (modes/schedules), as-built deviations and boundary conditions required for high-fidelity simulation.

Metamodeling Strategy. To keep the DM extensible we adopt an inheritance-and-interfaces strategy for modeling: a shallow class taxonomy captures stable building concepts (e.g., `MEP_Equipment`) with minimal specialization. Instead of deeper inheritance, `Ports` provide the compositional extensibility needed for uniform integration of both control signals and physical connectivity. Variability is thus handled via composition by introducing operational inputs through `Schedule` and reusable `Profiles`, which also drive a uniform representation of operational boundary conditions (e.g., comfort intervals for temperature) as `BoundaryCondition` objects. Optional capabilities are added via interface-based mixins (e.g., traceability) so that these concerns remain separated from the core entities.

Core Representation: BIM Entities and Port-enabled Connectivity. The DM provides a system-level representation of the building using a compact set of MBSE-aligned BIM entities (e.g., `Asset` and `MEP_Equipment`) that are semantically enriched for simulation. `Ports` are used for both control and physical domains, comprising `SignalPort` and `PhysicalPort`. `SignalPorts` represent measurable or controllable points (e.g., SP1 & 2 in Fig. 2a). Consequently, sensors and actuators are represented as signal-level endpoints associated with BIM elements, which enables explicit control-to-physical bindings. `PhysicalPorts` define physical interfaces with boundary variables that become part of the equation system (for simulation) upon connection. The physical topology is captured explicitly by `Connectors` which compose two `PhysicalPort`.

Operational and contextual semantics. The DM models operational and contextual semantics as inputs to simulation. Operational intent is represented by `OperatingSchedules` and `OperatingModes`: schedules select modes over time, and modes define the active set of operational inputs for a given trade (e.g., HVAC, lighting). Within a mode, operational inputs are expressed as `SetPoints` (e.g., T_{set} in Fig. 2b) that target a specific `SignalPort` and are parameterized by a `Profile`, which represents either constant values or time series via an XOR constraint and supports periodic or absolute time bases. External inputs are represented as `BoundaryConditions`. Each condition targets a `PhysicalPort`, such as a thermal or fluid boundary, and specifies the affected quantity and boundary type, such as temperature (e.g., T_{outdoor} (cf. Fig. 2b). The `PhysicalPort`'s input is provided via a `Profile`, which also supplies boundary-condition inputs.

A concrete simulation scenario is encapsulated by the `RunScenario`, which represents the entry point for a simulation run: it specifies the `EntryModel`, set of controllers to be simulated, operational schedules to apply, and boundary-condition inputs that define the simulation context. At the system level, the `RunScenario` orchestrates control by aggregating multiple `Controller` instances for modular composition of their control behavior. Execution details that are solver- or tool-specific are referenced externally (via `executionConfigUri`).

Traceability: Requirements and As-built Deviations. To support verification and lifecycle feedback, the DM provides two capability interfaces as traceability. `IAsBuiltDeviation` captures post-construction deviations in both building fabric (e.g., material thermal conductivity Δ_k or wall thickness Δ_{th} in Fig. 2a) and positioning (e.g., Δ_{pos} in Fig. 2a). This allows the simulation model to evolve from an idealized "As-Designed" state to a calibrated "As-Built" reality. Complementary to this, `IHasRequirement` embeds requirements as executable verification constraints (e.g., $\Delta T_{\text{overshoot}} \leq 1°\text{C}$ in Fig. 2c) into components to enable automated continuous compliance checking. Traceability eventually is realized by attaching such constraints and deviation records at appropriate scopes (e.g., scenario/zone/component), and by retaining identifiers, timestamps, and provenance data so that requirements, as-built/commissioning updates, and resulting analyses can be correlated across simulation runs and lifecycle iterations.

4.1.2 CLM: Formalized Control Logic Structure. The CLM (cf. Fig. 4) operates atop the DM and focuses on a structured definition of the control logic and signal flow for individual controllers.

Metamodeling Strategy. The CLM formalizes BCS control logic via typed, directional `ControlPorts`, while executable semantics are captured in a Modelica-aligned representation (cf. `ModelicaBody`) that separates variable and function declarations, equations, and algorithmic control-flow statements. The `ControlLogicModel` together with the `LogicBlock` supports reuse and composition of controller logic.

Core Elements: Logic Blocks and Ports. The CLM is structured around a block-oriented signal flow paradigm [43]:

`LogicBlock` The atomic unit of behavior, encapsulating a control function. Subtypes include:

- `LibraryBlock`: references and wraps standard Modelica components (e.g., `Buildings.Controls.PID`).
- `CustomBlock`: Encapsulates user-defined control logic. It utilizes a structured internal `ModelicaBody` that aligns with the Modelica specification [21]. This structural mapping ensures full syntactic fidelity.
- `StateMachineBlock`: Manages discrete system states and transitions (e.g., `Off` → `Startup` → `Running`), with support for `onEntry`, `onExit`, and `during` actions.

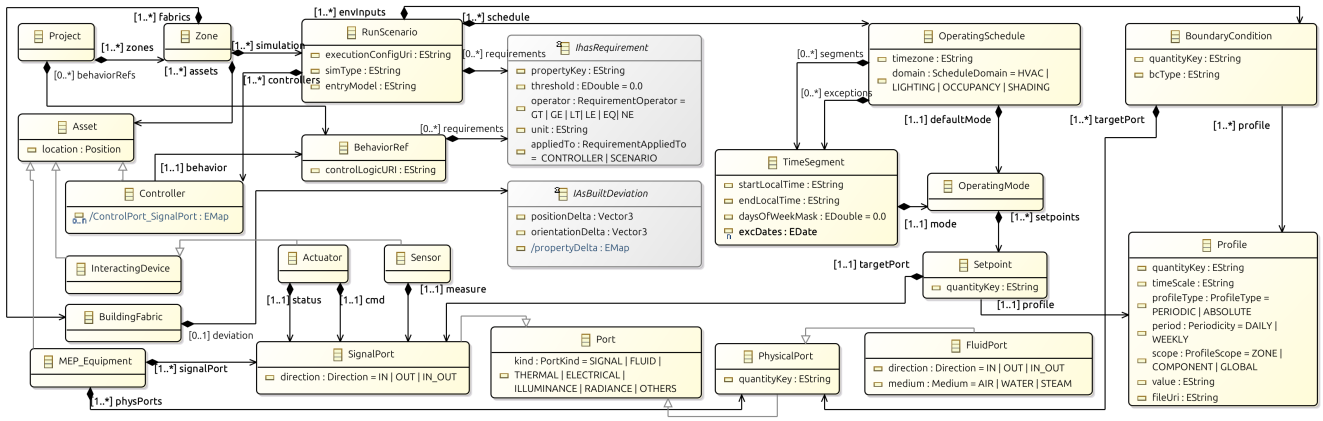


Figure 3: Domain metamodel. Best viewed zoomed in on a computer display.

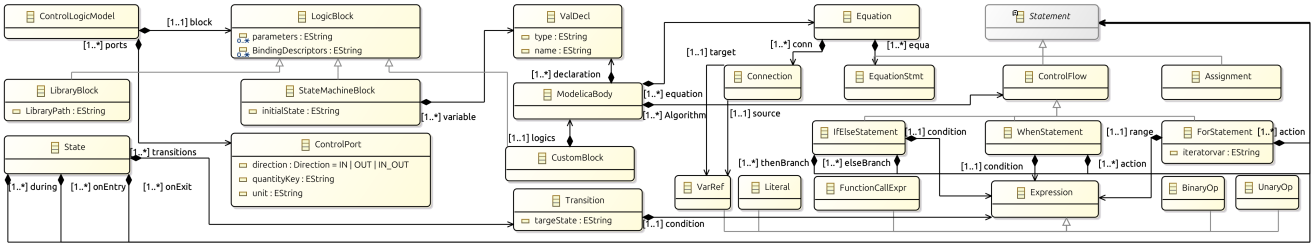


Figure 4: Control logic metamodel. Best viewed zoomed in on a computer display.

`ControlPort` Typed, directional interfaces that ensure signal compatibility through explicit dataTypes and associated semantics such as `quantityKey` or `unit`.

Connections and Algorithmic Constructs. While `ControlPorts` define the external interface of controller logic, internal wiring is captured in the `Equation` part of `ModelicaBody`. Specifically, an `Equation` contains `Connection` instances, where each links a pair of `VarRefs` (source and target), making connections explicit between port-associated variables and locally declared variables. In addition, the `CLM` defines executable statement and expression constructs for algorithmic behavior: control flow is modeled using guarded statements (e.g., `if/else`, `when`, `for`) and assignments, and expressions are built from variable references, literals, unary/binary operators, and function calls.

4.1.3 Integrating the DM with the CLM. The integration between the DM and the CLM is explicitly managed through controller configurations. As the primary agent of active logic within the system, the `Controller` *intrinsically* inherits from `Asset`, grounding it as a concrete building component. Crucially, it employs a *semantic indirection* [10] by the `BehaviorRef`, which decouples the physical entity from the digital control strategy. Specifically, the `BehaviorRef` defers the specific control logic to the CLM, such that the control strategy can be updated or swapped (e.g., switching from a PID to a custom controller via `BehaviorRef` in Fig. 2c) through simple (re)modeling.

In addition, the DM captures controller's interface configurations through explicit port bindings. These bindings map `ControlPorts` (cf. Fig. 4) to concrete `SignalPorts` in the building model (e.g., sensor measurements, setpoint inputs, and actuator command/feedback points) by "wiring" the controller interface to the controlled physical entity. Together, `BehaviorRef` and port bindings make controller configuration explicit and checkable at the DM level, while keeping controller-internal logic confined to the CLM.

4.2 Modeling Methodology

Building upon the metamodels from Section 4.1, we first operationalize this approach using *Eclipse Ecore* [19] to define the abstract syntax of our DSMLs. Its concrete syntax, representing the graphical components directly manipulated by users, is implemented using Revit Families. These Families are automatically generated from the Ecore metamodel via ATL [18].

Capitalizing on this tooling foundation, this section details the modeling methodology employed to transform design models into semantically enriched artifacts ready for simulation model generation. Specifically, the scope of this methodology covers the preconfiguration of the BCS within BIM models, which includes the definition of devices and their connections. To systematically achieve this transformation and ensure data consistency, our methodology proceeds in five sequential steps:

Step 1: Design Model Validation. The workflow begins with a standard BIM model. We define strict prerequisites for this design

model: it must include spatial zones, accurately modeled building fabrics, and precise material properties relevant to the target physical simulations. To guarantee simulation readiness and prevent errors such as missing semantic attributes or topologically unclosed zones, a model checker [58] verifies the presence and consistency of these attributes.

Step 2: Simulation Readiness. Once the model is validated, we address the simulation's execution requirements. We apply the DM to inject static simulation parameters. First, we instantiate a `RunScenario` to encapsulate execution configurations comprising `OperatingSchedules`, which define operational modes over time segments, and `BoundaryConditions`, which provide trade-specific physical inputs. The `RunScenario` also includes `IHasRequirements` to define KPIs for simulation evaluation. Next, we instantiate building control `Assets` (e.g., `Controllers`, `Sensors`, `Actuators`) and assign them to relevant `Zones`. These assets are linked to existing BIM model elements (e.g., walls, windows, ...) via globally unique identifiers (GUID). We then configure the semantic interfaces of these devices by defining corresponding `Ports` (e.g., `SignalPort`, `PhysicalPort`) to establish accurate topological connectivity. Finally, we implement `IAsBuiltDeviation` to record construction deviations and align the model with as-built realities. Coupling the static design with operational logic yields a BIM model ready for simulation model generation.

Step 3: Control Logic Specification. In this step, control scripts are specified by adhering to the syntax specifications defined by the CLM. Listing 1 provides an example of this modeling step, illustrating a `ControlLogicModel` (Lighting_Controller in Fig. 2b), which defines a proportional illuminance controller with safety thresholds.

Step 4: Automated Generation of Simulation Models. With the fully prepared model, the automated code synthesis framework processes the enriched model, systematically translating the explicitly defined physical parameters and control logic into executable Modelica models (cf. Section 4.3).

Step 5: Simulation Execution and Data Generation. Usually conducted during the planning and design phase of a building, this step leverages the generated models to both assess the building's design which at the same time produces the necessary high fidelity data for training surrogate models. This requires a systematic approach to define the number of necessary simulation runs. To achieve this, we employ Latin Hypercube Sampling (LHS) [1]. For this, we leverage the KPIs modeled as part of the BIM model (cf. `IHasRequirement`) to guide the identification of the relevant input parameters and boundary conditions that must be varied to capture the building's dynamics and modeling uncertainties comprehensively [28] (e.g., unmeasured air infiltration rates, stochastic internal heat gains from occupants, or actual versus nominal equipment efficiencies). By applying LHS to these variables of interest, we can iteratively generate different simulation configurations and identify the necessary number of simulations required for the surrogate model's predictive accuracy to converge.

Listing 1: YAML definition instantiating a CLM

```yaml
ControlLogicModel:
 entityID: "CTRL_001"
 modelName: "SmartLightingCtrl"
 ports:
  - {name: "lux_sensor", direction: "In", dataType: "Real", unit: "lux"}
  - {name: "dimmer_act", direction: "Out", dataType: "Real", unit: "%"}
 block:
  CustomBlock:
   parameters:
    - {name: "target", type: "Real", value: 500.0}
    - {name: "safety_threshold", type: "Real", value: 1200.0}
   logics:
    Declaration:
     - {type: "Real", name: "error"}
     - {type: "Boolean", name: "safety_mode"}
    Equation:
     - {id: "calc_err", content: "error=target-lux_sensor"}
    Algorithm:
     - IfElseStatement:
        condition: "lux_sensor>safety_threshold"
        thenBranch:
         - Assignment: {id: "set_safe", content: "safety_mode=true"}
         - Assignment: {id: "disable_out", content: "dimmer_act=0.0"}
        elseBranch:
         - Assignment: {id: "set_normal", content: "safety_mode=false"}
         - Assignment: {id: "pi_control", content: "dimmer_act=error*0.5"}
```

Our five-step modeling methodology ensures correct-by-construction generation [51] of simulation-ready Modelica models from MBSE-aligned BIM models and systematically captures system uncertainties via LHS to generate high-quality training data for robust surrogate models.

4.3 Simulation Model Generation

Our model generator uses the Industry Foundation Classes (IFC), an open standard for BIM model exchange that can readily be exported from common BIM authoring tools (e.g., Revit) as input. In the following, we outline its main components and the underlying model generation algorithm.

4.3.1 BIM2Modelica Mappings. To tackle the heterogeneity between the BIM and Modelica, our generator employs a knowledge base providing two mappings:

(1) Component Type Mapping (CTM). The CTM handles the semantic translation from BIM entities to Modelica classes. It provides a static dictionary that defines:

- Class Mapping: Maps `IfcEntity` and `PredefinedType` (e.g., `IfcFan.CENTRIFUGAL`) to a concrete Modelica class (e.g., `Buildings.Fluid.Movers.FlowControlled_m_flow`).
- Parameter Alignment: Explicitly maps BIM properties (e.g., `FlowRate`) to Modelica parameters (e.g., `m_flow_nominal`), thereby verifying data types and handling unit conversions.

(2) Interface & Port Mapping (IPM). The IPM handles the topological translation of connection interfaces. It acts as an interface adapter that normalizes generic BIM port definitions (e.g., `INLET`, `ControlInput`) into Modelica port identifiers (e.g., `port_a`, `u`, `y`). This ensures that both physical connections and logical signal pathways are correctly synthesized according to the Modelica's specification.

4.3.2 Synthesis Algorithm. The automated BIM-to-Modelica transformation is outlined in Algorithm 1 and takes three inputs: $\mathcal{I}$ (the BIM model in IFC), $\mathcal{M}_{comp}$ (CPM), and $\mathcal{M}_{port}$ (IPM) and operates in four main stages to construct a semantic graph $G(V, E)$ of the BCS topology and its logic for then generating simulation ready Modelica models:

Stage 1: Node Extraction (Lines 4–15). This stage systematically extracts and categorizes relevant building components to populate the base nodes of the target semantic graph G. Specifically, stage 1 parses $\mathcal{I}$ to instantiate graph vertices (i.e., nodes; V) in G.

Algorithm 1 BIM-to-Modelica Code Synthesis Algorithm

1: **procedure** SynthesizeModelica($\mathcal{I}$, $\mathcal{M}_{comp}$, $\mathcal{M}_{port}$)
2: $G \leftarrow (\emptyset, \emptyset)$
3: ▷ **Stage 1: Node Extraction and Resolution**
4: $E_{raw} \leftarrow$ ParseEntities($\mathcal{I}$)
5: **for** each entity $e \in E_{raw}$ **do**
6: $v \leftarrow$ CreateNode(ResolveType(e, $\mathcal{M}_{comp}$), ExtractAttributes(e, Rules))
7: **if** isPhysical(e) **then**
8: $V_{phys} \leftarrow V_{phys} \cup \{v\}$
9: **else if** isControl(e) **then**
10: $V_{log} \leftarrow V_{log} \cup \{v\}$
11: **else**
12: $V_{space} \leftarrow V_{space} \cup \{v\}$
13: **end if**
14: $V \leftarrow V \cup \{v\}$ ▷ Cache containment & connectivity pointers
15: **end for**
16: ▷ **Stage 2: Topology Reconstruction**
17: **for** each node $v \in V_{phys} \cup V_{log}$ **do**
18: ▷ Physical Topology Synthesis
19: **for** each port $p \in v.PhysicalPorts$ **do**
20: $target \leftarrow$ TraceConnectivity(p, $\mathcal{I}$)
21: $(p', target') \leftarrow$ ResolvePorts($p, target, \mathcal{M}_{port}$)
22: **if** Compatible($p', target'$) **then**
23: $E \leftarrow E \cup$ CreateConnector($p', target'$)
24: **end if**
25: **end for**
26: ▷ Control Wiring Resolution
27: **if** $v \in V_{log}$ **then**
28: **for** each binding $(p_{ctrl}, p_{signal}) \in v$.GetPortBindings() **do**
29: $E \leftarrow E \cup$ CreateSignalEdge(p_{ctrl}, p_{signal})
30: **end for**
31: **end if**
32: **end for**
33: ▷ **Stage 3: Spatial Context and Physics Interaction**
34: **for** each node $v \in V_{phys}$ **do**
35: $zone \leftarrow$ GetSpatialParent(v, $\mathcal{I}$)
36: **if** $zone \neq$ null **then**
37: $ctx \leftarrow$ DeterminePhysicsContext(v)
38: $E \leftarrow E \cup$CreateIntLink(v.GetAmbPort(ctx), $zone$.GetEnvPort(ctx))
39: **end if**
40: **end for**
41: ▷ **Stage 4: Modelica Code Emission**
42: $C_{out} \leftarrow$ Initialize empty Modelica model text
43: ▷ Render component instantiations
44: **for** each node $v \in V$ **do**
45: C_{out}.append(RenderTemplate(v, ModelicaLibrary))
46: **end for**
47: ▷ Generate topology equations
48: **for** each edge $e = (src, tgt) \in E$ **do**
49: C_{out}.append(GenerateConnectEquation(src, tgt))
50: **end for**
51: **return** C_{out} ▷ Spit out the executable simulation model
52: **end procedure**

Using the CTM (M_{comp}), each relevant IFC entity is resolved to its corresponding Modelica component type. Nodes are categorized by their domain into V_{phys} (physical components), V_{log} (control-logic elements), or V_{space} (spatial elements), while caching essential spatial and topological pointers.

Stage 2: Topology Reconstruction (Lines 17–31). The algorithm reconstructs BCS connectivity by resolving physical interfaces through `PhysicalPorts`, thereby normalizing them via M_{port}, and instantiating compatible pairs as connector edges (Lines 19–25). It then resolves control bindings within controller nodes by mapping `ControlPorts` to the corresponding physical `SignalPorts` (Lines 27–30). Both relation types are instantiated as directed edges in E, thereby assembling connecting the nodes $V \in G$.

Stage 3: Spatial Context and Physics Interaction (Lines 33–39). The algorithm next identifies the parent spatial element node ($zone$) for each node in V_{phys} by querying its boundary definitions to delimit the specific spatial enclosure (Line 35). These spatial containments are then translated into interaction edges to link the node's interfaces to the environment (Lines 36–38).

Stage 4: Automated Code Synthesis (Lines 41–51). Finally, by traversing the fully populated graph $G(V, E)$, the algorithm injects parameters into predefined Modelica templates (Lines 44–46) and translates the synthesized edges into standard `connect()` equations (Lines 48–50) to output executable models.

Algorithm 1 guarantees the generation of simulation-ready Modelica models with time complexity $O(m \times n)$, where m is the total number of nodes V and n is the average node degree in G. This ensures high scalability.

4.4 Implementation Details

As discussed in Section 4.2, ATL is used for generating Revit Families. These artifacts require further processing to be functional for modeling within Revit. To address this, we developed a pyRevit plugin to embed these Families into Revit's modeling infrastructure. This application of model-based tool integration [35] transforms Revit into a BIM2SIM-ready design platform. This workflow of this tool (Fig. 5) operates as a linear pipeline that transforms geometric data into semantic objects: upon opening a BIM model in Revit, the plugin injects the aforesaid Families into the BIM model. The workflow relies on a semantic infrastructure (the middle part of Fig. 5) which consists of five components:

The Domain Schema The DM is transformed via ATL and serialized into a JSON schema using Acceleo [17] yielding the target Revit Families.

The Attributes Corpus To align with Revit's native parameter extension mechanism, the semantic attributes defined in the DM are materialized using a shared attribute corpus. This serves as the primary "vocabulary" for populating BIM entities, with each attribute definition indexed by a GUID.

The Family Type Extension Revit Family types are extended with DM-added types, such as `OperatingSchedule` and `BoundaryCondition`.

The Family Binding To achieve semantic enrichment, Revit Families within the active project are bound to multiple attributes in the shared corpus via GUIDs.

The Family Identification To ensure that a Family binding is established with the correct attribute mappings, the plugin resolves the identity of Revit Families through a tiered identification process:

(1) Intrinsic Classification: The approach leverages Revit's native `BuiltInCategory` property to resolve distinct architectural elements (e.g., mapping `OST_Walls` directly to `BuildingFabric`). For ambiguous categories such as `OST_MechanicalEquipment`, it queues them for precise type resolution in subsequent steps.
(2) Dictionary Lookup: It queries the loaded mappings to identify exact matches between Revit families and the metamodel types.
(3) Fallback: Acts as a safety net by analyzing keywords or triggering manual annotation input to get identity. Successful matches are merged back into the knowledge base.

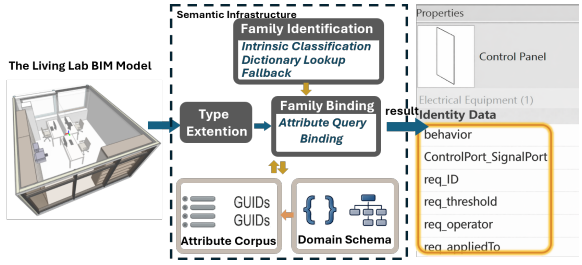


Figure 5: The Revit Representation Enrichment Approach

4.4.1 Code Synthesis. Our code generator is implemented in `Python` and orchestrates the model transformation (cf. Section 4.3). To parse the input BIM (IFC, respectively) model, it leverages `IfcOpenShell` [33] to traverse the IFC model, systematically extract structural properties, spatial definitions, and raw relational data, and reconstruct the BCS's full topology (cf. Algorithm 1). For more complex spatial interactions, such as boundary topologies, we utilize `NetworkX` [44] to construct a directed graph. Semantic model mappings and material datasets are preserved in external `YAML` and `JSON` configurations. Finally, the Modelica models are synthesized and emitted via `Jinja2`-based [45] templates.

5 Evaluation

We evaluate our proposed BIM2SIM workflow in a Living Lab at the University of Innsbruck, a 22m^2 office space, which is equipped with a BCS using DALI-based lighting and KNX-based shading control, and a multi-modal sensor infrastructure. We validate (i) the generated simulation models and (ii) the surrogate models against operational realities. To this end and to answer our **RQs** (cf. Section 1), we evaluate our proposal across three dimensions. Simulation accuracy is assessed by comparing synthesized data against Living Lab data using $\mathcal{M}$ [61] (an IoU-inspired measure for time-series alignment), R^2 [46] (coefficient of determination), and Pearson Correlation Coefficient (r) [37]. Automation efficiency is evaluated through a scalability analysis, measuring model generation times under varying model scale and complexity. Surrogate modeling is evaluated by measuring prediction accuracy using the aforesaid metrics ($\mathcal{M}$, R^2, and r) and comparing simulation execution times vs. model inference speed.

Experimental Setup: The BIM model of the Living Lab is exported in IFC4 from Revit 2026, after being enriched using the metamodels from Section 4.1 and the methodology from Section 4.2. Experiments were performed on a laptop running Ubuntu 22.04 LTS with an Intel Core i7-1365U CPU, 40 GB memory, and Python 3.10 for all implementations. The generated simulation models were executed in OpenModelica v1.25.5. For the accuracy evaluation, we used a validation dataset of historical BCS illuminance readings from our Living Lab with a 10-second sampling interval over a 102-day (July 16th-October 25th, 2025) high-dynamics period as operational baseline. We applied LHS (cf. Section 4) to sufficiently cover the parameter space of key variables during simulations. Specifically, we varied four parameters: Blind Closure Fraction (BCF, ranging from 0 to 1), Slat Angle Level (SAL, utilizing discrete states such as 0°, 30°, 45°, 60°, and 85°), Lamp Activation Probability (LAP, representing the per-5-minute turn-on probability during occupancy), and Lamp Deactivation Probability (LDP, representing the per-5-minute turn-off probability after turn-on). We iteratively generated simulations, starting from 30 simulation runs, and incrementally increasing the sample size until the surrogate model's training converged, resulting in 100 simulation runs.

5.1 Simulation Accuracy

As shown in Fig. 6, the generated simulation models accurately capture the building's daylight dynamics. The simulation achieves $R^2 = 0.69$ and $r = 0.84$, confirming that the general trends observed in the measured data are faithfully reproduced by the generated simulation models. Table 1a complements this view with summary accuracy metrics, where the simulation outputs achieve $\mathcal{M} = 0.59$. This performance is consistent with documented energy performance gaps, which typically range from 10% to 200% across building types [16].

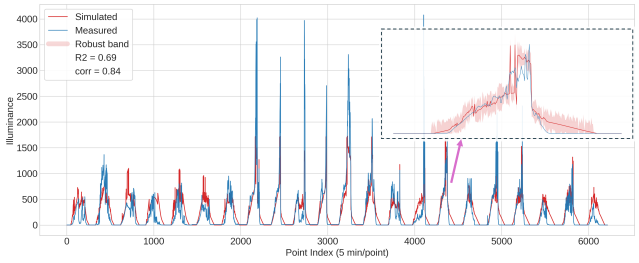


Figure 6: Simulation accuracy for the Living Lab study (measured vs. simulated illuminance).

5.2 Surrogate Modeling

We trained a Mamba-based surrogate model [25, 59] on the simulated dataset whose size we iteratively increased, such that the learned dynamics become more robust (cf. the *Robust band* in Fig. 6). The robustness of the resulting data is confirmed by Table 1b which shows summary statistics for $\mathcal{M}$ ranging from $\mathcal{M}_{\text{min}} = 0.51$ (worst) to $\mathcal{M}^* = 0.55$ (best), with a median of 0.52 and a mean of 0.53. These metrics align with documented performance gaps in building energy simulation [16]. Table 2 summarizes our results for surrogate model training from this data set and with the five best surrogate candidates and the best model ($\text{surro}_{\text{sltd}}$). $\text{surro}_{\text{sltd}}$ shows strong accuracy ($R^2 = 0.93$, $r = 0.96$, MAE = 31.88, RMSE = 93.85).

Table 1: Accuracy metrics between measured illuminance, simulation outputs, and surrogate predictions.

(a) Comparison against measured data.

Metric	Simulation	$\text{surro}_{\text{sltd}}$	$\text{surro}_{\text{cand5}}$	$\text{surro}_{\text{cand4}}$	$\text{surro}_{\text{cand1}}$
$\mathcal{M}$	**0.59**	**0.64**	0.63	0.63	0.63
R^2	**0.69**	**0.74**	0.72	0.72	0.73
r	**0.84**	**0.87**	0.86	0.86	0.87

(b) Summary statistics for data set robustness.

Model	$\mathcal{M}_{\text{min}}$	$\mathcal{M}_{\text{median}}$	$\mathcal{M}_{\text{mean}}$	$\mathcal{M}^*$
Simulation	0.51	0.52	0.53	0.55

Table 2: Surrogate-model performance on simulated data.

Model	N	T_{train} (s)	$\text{Epoch}_{\text{sel}}$	R^2	r	MAE	RMSE
$\text{surro}_{\text{cand1}}$	75	7941	70	0.93	0.96	31.28	89.57
$\text{surro}_{\text{cand2}}$	80	8853	100	0.93	0.96	30.82	91.99
$\text{surro}_{\text{cand3}}$	90	9157	70	0.92	0.96	32.89	95.68
$\text{surro}_{\text{cand4}}$	95	10057	80	0.93	0.96	29.50	88.51
$\text{surro}_{\text{cand5}}$	100	9998	100	**0.94**	**0.97**	**27.65**	**85.80**
$\text{surro}_{\text{sltd}}$	*85*	*9585*	*80*	*0.93*	*0.96*	*31.88*	*93.85*

Table 3: Scalability results (median over 50 runs, in secs).

(a) Number of controllers (N_C) with increasing nesting depth (D).

D vs. N_C	1	10	100	500
0	1.0354	1.0332	1.0444	1.1059
1	1.0356	1.0362	1.0500	1.1149
3	1.0669	1.0605	1.0704	1.1749
5	1.3045	1.3140	1.3226	1.4191

(b) Number of devices (N_D) with increasing port numbers (P).

N_D vs. P	R	1	2	5	10	15
1	50	1.0637	1.0626	1.0651	1.0653	1.0664
100	50	1.0727	1.0710	1.0710	1.0719	1.0764
500	50	1.1133	1.1167	1.1226	1.1210	1.1240
1000	50	1.2062	1.2078	1.2153	1.2122	1.2274

Against measured data, it achieves the best agreement ($\mathcal{M} = 0.64$, $R^2 = 0.74$, $r = 0.87$), as shown in Table 1a. This improvement over simulation (cf. Table 1a) is expected as the surrogate is trained across 100 simulation runs, each with different parameter configurations. It therefore learns a representation that implicitly averages over parameter uncertainties which can yield predictions closer to measured data than any single simulation run with fixed assumptions.

$\text{surro}_{\text{sltd}}$ was selected as the best model due to requiring fewer training samples, shorter training time, and fewer epochs while maintaining comparable accuracy (cf. $\text{surro}_{\text{cand5}}$). The generally short training times reinforce the feasibility of our approach for rapid and agile surrogate model deployment for DTs. Observe that Table 1a reports metrics on operational data, whereas Table 2 reports metrics on simulated data.

5.3 Scalability and Performance

We assess scalability and performance from two complementary perspectives: the cost of model generation as size and structural complexity increase, and the runtime difference between physics-based simulation and surrogate model inference. To this end, we created two benchmarks to stress-test the generation workflow. The first benchmark varies controller counts N_C and logic nesting depth D, whereas the second varies the number of devices N_D and the number of ports P per device. For the first benchmark (cf. Table 3a), the median execution time increases only from about 1.035 s to 1.419 s. The stepwise increase with nesting depth reflects the additional traversal cost of deeper control-logic trees, while the overall overhead remains low.

Table 3b shows the results for the second benchmark. As N_D scales up to 1000, the median execution time remains highly stable. Even when port numbers increase from $P = 1$ to $P = 15$, generation times increase only marginally. Under maximum tested load ($N_D = 1000$, $P = 15$), the median generation time is 1.23 s, confirming robust scalability even for large-scale BCS installations. These considerably short generation times enable rapid scenario exploration during simulation modeling, and efficient dataset generation for surrogate modeling.

Table 4: Inference-time of simulation vs. surrogate models.

Metric	Step	Simulation	$\text{surro}_{\text{sltd}}$
Cumulative time (s)	1	11.15	0.0012
Cumulative time (s)	100	1,019.27	0.13
Cumulative time (s)	500	5,215.05	0.62
Cumulative time (s)	1000	10,272.76	1.25
Cumulative time (s)	2000	20,172.69	2.50
Single-step time (ms)	Median	9,980.2	1.25
Single-step time (ms)	Mean	**10,086.3**	**1.25**

Table 4 shows the results for simulation duration vs. inference performance. The Radiance-OMC simulation requires around 20000 s for 2000 simulation steps, whereas the surrogate model completes in 2.50 s (Table 4). The mean single-step execution time drops from 10086.3 ms to 1.25 ms, demonstrating that our trained surrogate models can effectively support the fine-grained, near real-time predictions required for DT operations [32].

Our results demonstrate end-to-end efficiency of our model-driven pipeline: fast model generation enables rapid scenario exploration and large-scale dataset creation for surrogate modeling. While absolute execution times may vary across runtime conditions [15], performance and scalability trends remain consistent, confirming that model-driven automation directly enables practical DT applications.

6 Discussion

Our evaluation demonstrates that model-driven automation successfully addresses the semantic gaps and media breaks in current BIM2SIM workflows. The MBSE-aligned metamodels and deterministic transformations preserve the semantic fidelity needed for operational simulations, achieving strong correlation with measured data ($\mathcal{M} = 0.59$, $R^2 = 0.69$, $r = 0.84$, cf. Table 1a) while maintaining efficient generation times (under 1.5s for complex models, cf. Table 3). This validates that MDE techniques can effectively bridge the gap between static design models and dynamic behavioral simulation. The simulation accuracy validates our five-step modeling methodology. The explicit port bindings and controller configurations captured during Steps 2-3 (cf. Section 4.2) translate directly into correct Modelica components, demonstrating that correctness-by-construction holds across the full pipeline. This end-to-end consistency eliminates the manual rework and semantic inconsistencies that plague traditional BIM2SIM approaches.

The surrogate models trained on simulation-generated data perform even better against operational data ($\mathcal{M} = 0.62$, $R^2 = 0.74$, $r = 0.87$, cf. Table 1a). More importantly, the automated pipeline enables systematic dataset generation at scale, with a four-orders-of-magnitude speedup (20,173s to 2.5s for 2000 steps) that directly enables practical DT applications. The scalability results confirm that the model-driven pipeline scales efficiently with model complexity. Generation times remain under 1.5 seconds even for models with 500 controllers at nesting depth 5, or 1000 devices with 15

ports each. This efficiency stems from the structured metamodeling and model transformations, both core MDE principles. The short training times for surrogate models further demonstrates how our approach facilitates rapid, agile, and iterative DT-based optimization.

Our **RQs** asked what semantic structures enable behavioral modeling in BIM (**RQ1**) and how to automate their translation to simulation (**RQ2**). The results show these questions are interdependent: explicit port-based interfaces, controller behavior references, and operational schedules only become effective when combined with deterministic model transformations and graph-based topology reconstruction. The semantic structures must be designed for automated transformation, not just manual comprehension. Our evaluation confirms that this integrated approach provides sufficient semantic expressiveness while enabling efficient automation.

6.1 Implications and Lessons Learned

Our work demonstrates how MDE principles address semantic heterogeneity in complex cyber-physical systems (CPS) where design, control, and runtime concerns intersect across multiple toolchains. The key contribution for the MDE community is a systematic approach to bridging tool-induced media breaks through explicit metamodeling of cross-domain semantics. We found that compositional extensibility through ports and interfaces scales better than deep inheritance hierarchies when modeling heterogeneous technical systems. This approach enables uniform integration of disparate concerns without proliferating specialized types. The lesson is that keeping metamodels shallow and extensible through composition maintains both domain flexibility and automated transformation viability.

The model transformations eliminated manual translation steps that typically introduce errors. The lesson for MDE is that automation value comes from end-to-end traceability: when semantic structures align across design, transformation, and execution phases, engineers can iterate rapidly without rework. Graph-based topology reconstruction proved robust for handling bidirectional physical and control relationships. Template-based code synthesis from well-formed metamodel instances realizes correctness-by-construction, demonstrating that MDE can provide strong guarantees even when targeting flexible languages like Modelica. The key is tight alignment between metamodel semantics and target language constructs.

Our approach shows that MDE can make existing domain standards (IFC, BIM) interoperable with simulation platforms through semantic enrichment layers rather than wholesale replacement. This suggests a practical MDE adoption path in established industries: augment existing artifacts and automate practitioner workflows. We found that evaluating generated artifacts against real operational data provides stronger validation than synthetic benchmarks alone. For MDE research targeting CPSs, this emphasizes the importance of empirical validation in realistic settings where model fidelity matters for downstream use.

Practically, this work demonstrates that model-driven approaches can scale beyond software systems to building-scale cyber-physical infrastructure by handling the semantic complexity and scale requirements of real-world DT applications. We encountered several challenges: (i) IFC's limited behavioral semantics required metamodel extensions, (ii) model transformations required careful topology reconstruction, (iii) simulation accuracy depends on accurate boundary conditions, where small input deviations propagate through models, and (iv), while model transformation enables correctness-by-construction, target language flexibility merits edge case validation.

6.2 Limitations and Threats to Validity

Despite demonstrating significant automation potential, this study has several limitations:

Scope Limitation. The current implementation is restricted to illuminance. Complex multi-physics interactions, such as HVAC systems, have not been evaluated.

External Validity. Our validation relies on a single Living Lab case study. Generalizability to buildings of varying scales and topological complexities requires further empirical investigation.

Internal Validity. Execution times in Tables 2 to 4 are subject to hardware-specific variations. Relative performance trends however remain consistent across runs.

Construct Validity. We evaluate simulation outputs using trend analysis. Physics-based simulations inherently deviate from reality due to unpredictable occupant behavior and imprecise conditions, known as the energy performance gap [52]. Capturing behavioral trends sufficiently demonstrates validity.

Simulation Viability. Generated models may occasionally fail due to Modelica's flexibility. Our framework mitigates this through model transformations, which realize "correctness by construction" [12, 36].

Future work will integrate HVAC systems to enable cross-trade simulations and explore modular model generation to reduce computational burden and improve scalability for large projects. We will also conduct user studies using the Technology Acceptance Model to assess the usability, usefulness, and adoption potential of our proposal in realistic BIM2SIM settings.

7 Conclusion

In this paper, we proposed an approach that integrates MDE, specifically, MBSE, with BIM to establish a model-driven pipeline from BIM to executable simulation models. Our work addresses persistent media breaks in current BIM2SIM workflows and provides an explicit methodology for carrying domain semantics and traceability information into downstream simulation artifacts. Our evaluation results indicate that our proposal is both technically feasible and practically relevant. The generated simulation models reproduce meaningful behavioral trends, while the model-driven pipeline remains efficient across varying model size and complexity. In addition, the synthetic data generated during simulation were used to train a surrogate model, whose predictions capture regular building behavior. This demonstrates that our model-driven pipeline can be leveraged to train a surrogate model for DTs. Taken together, our findings suggest that the approach can reduce manual remodeling effort, improve consistency between design and simulation artifacts, and support the scalable generation of high-fidelity synthetic data for downstream surrogate modeling.

Acknowledgments

This research has received funding from the Austrian Research Promotion Agency (FFG) under Grant Agreement No.: 927874, ELEVATE.

References

[1] Arshad Afzal, Kwang-Yong Kim, and Jae-Won Seo. 2017. Effects of Latin hypercube sampling on surrogate modeling and optimization. *International Journal of Fluid Machinery and Systems* 10 (02 2017). doi:10.5293/IJFMS.2017.10.3.240

[2] Mohammed Alhammad, Matt Eames, and Raffaele Vinai. 2024. Enhancing Building Energy Efficiency through Building Information Modeling (BIM) and Building Energy Modeling (BEM) Integration: A Systematic Review. *Buildings*. 14, 3 (2024).

[3] Ando Andriamamonjy, Dirk Saelens, and Ralf Klein. 2018. An automated IFC-based workflow for building energy performance simulation with Modelica. *Automation in Construction* 91 (2018), 166–181. doi:10.1016/j.autcon.2018.03.019

[4] Autodesk, Inc. 2025. Autodesk. https://www.autodesk.com/. Accessed: 2026-03-09.

[5] Salman Azhar. 2011. Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadership and Management in Engineering* 11 (07 2011), 241–252. doi:10.1061/(ASCE)LM.1943-5630.0000127

[6] Isa Banagar, Iiso Kramsu, Juho Könnö, and Amin Andwari. 2025. Connecting Systems Engineering and Domain Expertise through a Toolchain: A Simulation-Integrated MBSE Approach. *Journal of Systems and Software* 232 (10 2025), 112671. doi:10.1016/j.jss.2025.112671

[7] Gabriela Bastos Porsani, Kattalin Del Valle de Lersundi, Ana Sánchez-Ostiz Gutiérrez, and Carlos Fernández Bandera. 2021. Interoperability between Building Information Modelling (BIM) and Building Energy Model (BEM). *Applied Sciences* 11, 5 (2021). doi:10.3390/app11052167

[8] Paul Beery and Eugene Paulo. 2019. Application of Model-Based Systems Engineering Concepts to Support Mission Engineering. *Systems* 7, 3 (2019). doi:10.3390/systems7030044

[9] Maximilian Billmaier, Jounes-Alexander Gross, and Elias Bader. 2024. MBSE and BIM: Interdisciplinary Strategies for Enhancing Value in the Digital Transformation of the AEC Industry. https://conf.researchr.org/details/models-2024/models-2024-industry-day/8/MBSE-and-BIM-Interdisciplinary-Strategies-for-Enhancing-Value-in-the-Digital-Transfo. MODELS 2024 Industry Day, Accessed: 2026-03-09.

[10] Birthe Böhm, Norbert Gewald, Gerold Herold, and Dieter Wißmann. 2007. Indirection Pattern for data modeling. In *Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP)*. https://api.semanticscholar.org/CorpusID:6922431 Writers' Workshop D1.

[11] Nikolai Bolshakov, Xeniya Rakova, Alberto Celani, and Vladimir Badenko. 2023. Operation Principles of the Industrial Facility Infrastructures Using Building Information Modeling (BIM) Technology in Conjunction with Model-Based System Engineering (MBSE). *Applied Sciences* 13, 21 (2023). doi:10.3390/app132111804

[12] Tabea Bordis, Tobias Runge, Alexander Kittelmann, and Ina Schaefer. 2023. Correctness-by-Construction: An Overview of the CorC Ecosystem. *Ada Lett.* 42, 2 (April 2023), 75–78. doi:10.1145/3591335.3591343

[13] Ágnes Bárkányi, Tibor Chován, Sándor Németh, and János Abonyi. 2021. Modelling for Digital Twins—Potential Role of Surrogate Models. *Processes* 9, 3 (2021). https://www.mdpi.com/2227-9717/9/3/476

[14] Annamaria Ciccozzi, Andrea Santavicca, Tullio de Rubeis, and Dario Ambrosini. 2025. BIM-BEM interoperability for energy analysis: A comparative study of different strategies. *Energy Reports* 13 (2025), 4705–4718. doi:10.1016/j.egyr.2025.04.027

[15] Daniel Bristot de Oliveira, Daniel Casini, and Tommaso Cucinotta. 2023. Operating System Noise in the Linux Kernel. *IEEE Trans. Comput.* 72, 1 (2023), 196–207. doi:10.1109/TC.2022.3187351

[16] Shoumik Desai, Burak Gunay, Mohamed Ouf, Weihao Liu, and Joe Coady. 2026. Rethinking building energy performance gap: a workflow for bridging measured, modeled, designed, and optimized strategies–a review with real-world examples. *Energy and Buildings* (2026), 117065.

[17] Eclipse Foundation. 2025. Acceleo Project. https://projects.eclipse.org/projects/modeling.acceleo.

[18] Eclipse Foundation. 2025. ATL - a model transformation technology. https://eclipse.dev/atl/.

[19] Eclipse Foundation. 2025. Ecore Tools Overview. https://eclipse.dev/ecoretools/overview.html.

[20] Martin Fowler. 2010. *Domain-specific languages*. Pearson Education.

[21] Peter Fritzson and Vadim Engelson. 1998. Modelica—A unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*. Springer, 67–90.

[22] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. 2020. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access* 8 (2020), 108952–108971. doi:10.1109/access.2020.2998358

[23] Georgios Gourlis and Iva Kovacic. 2022. A holistic digital twin simulation framework for industrial facilities: BIM-based data acquisition for building energy modeling. *Frontiers in Built Environment* Volume 8 - 2022 (2022). doi:10.3389/fbuil.2022.918821

[24] Pascal Graignic, Thomas Vosgien, Marija Jankovic, Vincent Tuloup, Jennifer Berquet, and Nadège Troussier. 2013. Complex System Simulation: Proposition of a MBSE Framework for Design-Analysis Integration. *Procedia Computer Science* 16 (2013), 59–68. doi:10.1016/j.procs.2013.01.007 2013 Conference on Systems Engineering Research.

[25] Albert Gu and Tri Dao. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *First Conference on Language Modeling*. https://openreview.net/forum?id=tEYskw1VY2

[26] Pengfei Gu, Zhen Chen, Lin Zhang, Yuteng Zhang, Kunyu Xie, Chun Zhao, Fei Ye, and Yiran Tao. 2024. X-SEM: A modeling and simulation-based system engineering methodology. *Journal of Manufacturing Systems* 74 (2024), 198–221. doi:10.1016/j.jmsy.2024.01.013

[27] Sebastian Götz, Andreas Fehn, Frank Rohde, and Thomas Kühn. 2020. Model-driven Software Engineering for Construction Engineering: Quo Vadis? *The Journal of Object Technology* 19 (01 2020), 2:1. doi:10.5381/jot.2020.19.2.a2

[28] Sascha Hammes, David Geisler-Moroder, Johannes Weninger, Philipp Zech, and Rainer Pfluger. 2024. Market demands vs. scientific realities: A comparative analysis in the context of BIM-based and user-centred lighting control. *Developments in the Built Environment* 19 (2024), 100526. doi:10.1016/j.dibe.2024.100526

[29] Sascha Hammes, Johannes Strohm, David Geisler-Moroder, Johannes Weninger, Josef Miller, Martin Hauer, Philipp Zech, Mathias Wambsganß, and Rainer Pfluger. 2026. Technical and organizational barriers and solution strategies for the implementation of BIM: An empirical study with a focus on the lighting sector. *Results in Engineering* 29 (2026), 108520. doi:10.1016/j.rineng.2025.108520

[30] Martin Hauer, Sascha Hammes, Philipp Zech, David Geisler-Moroder, Daniel Plörer, Josef Miller, Vincent van Karsbergen, and Rainer Pfluger. 2024. Integrating digital twins with BIM for enhanced building control strategies: A systematic literature review focusing on daylight and artificial lighting systems. *Buildings* 14, 3 (2024), 805.

[31] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28 (03 2004), 75–105.

[32] Weifei Hu, Jiquan Yan, Feng Zhao, Chen Jiang, Hongwei Liu, Hyunkyoo Cho, and Ikjin Lee. 2023. Surrogate-based time-dependent reliability analysis for a digital twin. *Journal of Mechanical Design* 145, 9 (2023), 091708.

[33] IfcOpenShell Contributors. 2025. IfcOpenShell. https://ifcopenshell.org/.

[34] David Jansen, Philipp Mehrfeld, Dirk Müller, Eric Fichter, Veronika Richter, Andre Barz, Jesse Brunkhorst, Mike Dahncke, Pooyan Jahangiri, Christian Warnecke, Philipp Mehrfeld, Jérôme Frisch, Christoph van Treeck, and Bruno Lüdemann. 2021. BIM2SIM - Development of semi-automated methods for the generation of simulation models using Building Information Modeling. In *Proceedings of Building Simulation 2021: 17th Conference of IBPSA (Building Simulation, Vol. 17)*. IBPSA, Bruges, Belgium, 1244–1245. doi:10.26868/25222708.2021.30228

[35] Gerti Kappel, Manuel Wimmer, Werner Retschitzegger, and Wieland Schwinger. 2011. *Leveraging Model-Based Tool Integration by Conceptual Modeling Techniques*. Springer Berlin Heidelberg, Berlin, Heidelberg, 254–284. doi:10.1007/978-3-642-17505-3_12

[36] Mounira Kezadri Hamiaz, Marc Pantel, Xavier Thirioux, and Benoit Combemale. 2016. Correct-by-construction model driven engineering composition operators. *Form. Asp. Comput.* 28, 3 (May 2016), 409–440. doi:10.1007/s00165-016-0354-6

[37] Wilhelm Kirch. 2008. *Pearson's Correlation Coefficient*. Springer Netherlands, Dordrecht, 1090–1091. doi:10.1007/978-1-4020-5614-7_2569

[38] ByungChan Kong and WoonSeong Jeong. 2025. Development of an FMI-Based Data Model to Support a BIM-Integrated Building Performance Analysis Framework. *Buildings* 15, 17 (2025). doi:10.3390/buildings15173200

[39] Jie Li, Lisa Iulo, and Ute Poerschke. 2023. A Review of the Energy Performance Gap between Predicted and Actual Use in Buildings. In *Proceedings of Building Simulation 2023: 18th Conference of IBPSA*. 3406–3413. doi:10.26868/25222708.2023.1430

[40] Wenhui Liu, Yihan Lv, Qian Wang, Bo Sun, and Dongchen Han. 2024. A Systematic Review of the Digital Twin Technology in Buildings, Landscape and Urban Environment from 2018 to 2024. *Buildings* 14, 11 (2024). doi:10.3390/buildings14113475

[41] Modelica Association. 2025. Modelica. https://modelica.org/. Accessed: 2026-03-09.

[42] Joseph Murphy, Siyuan Ji, Charles Dickerson, Chris Goodier, Sonia Zahiroddiny, and Tony Thorpe. 2025. When BIM Meets MBSE: Building a Semantic Bridge for Infrastructure Data Integration. *Systems* 13, 9 (2025). doi:10.3390/systems13090770

[43] Sandeep Neema, Zsolt Kalmar, Feng Shi, Attila Vizhanyo, and Gabor Karsai. 2005. A visually-specified code generator for simulink/stateflow. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. IEEE, 275–277.

[44] NetworkX Developers. 2025. NetworkX. https://networkx.org/.

[45] Pallets Projects. 2025. Jinja2. https://pypi.org/project/Jinja2/.

[46] David A Pierce. 1979. R 2 measures for time series. *J. Amer. Statist. Assoc.* 74, 368 (1979), 901–910.
[47] Ana Luísa Ramos, José Ferreira, and Jaume Barcelo. 2012. Model-Based Systems Engineering: An Emerging Approach for Modern Systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42 (01 2012), 101–111. doi:10.1109/TSMCC.2011.2106495
[48] Peter Remmen, Jun Cao, Ebertshäuser Sebastian, Jérôme Frisch, Moritz Lauster, Tobias Maile, James O'Donnell, Pinheiro Sergio, Jörg Rädler, Rita Streblow, Matthis Thorade, and Reinhard Wimmer. 2015. An Open Framework for Integrated Bim-Based Building Performance Simulation using Modelica. In *Proceedings of Building Simulation 2015: 14th Conference of IBPSA (Building Simulation, Vol. 14)*. IBPSA, Hyderabad, India, 379–386. doi:10.26868/25222708.2015.2384
[49] Andrea Revolti, Luca Gualtieri, Pieter Pauwels, and Patrick Dallasega. 2024. From building information modeling to construction digital twin: a conceptual framework. *Production & Manufacturing Research* 12 (08 2024). doi:10.1080/21693277.2024.2387679
[50] Marcus Sandberg, Jani Mukkavaara, Farshid Shadram, and Thomas Olofsson. 2019. Multidisciplinary Optimization of Life-Cycle Energy and Cost Using a BIM-Based Master Model. *Sustainability* 11, 1 (2019). doi:10.3390/su11010286
[51] D.C. Schmidt. 2006. Guest Editor's Introduction: Model-Driven Engineering. *Computer* 39, 2 (2006), 25–31. doi:10.1109/MC.2006.58
[52] Xing Shi, Binghui Si, Jiangshan Zhao, Zhichao Tian, Chao Wang, Xing Jin, and Xin Zhou. 2019. Magnitude, Causes, and Solutions of the Performance Gap of Buildings: A Review. *Sustainability* 11, 3 (2019). doi:10.3390/su11030937
[53] Carlos Eduardo Gomes de Souza, Christine Kowal Chinelli, Carlos Alberto Pereira Soares, and Orlando Celso Longo. 2025. BIM as a Tool for Developing Smart Buildings in Smart Cities: Potentialities and Challenges. *Architecture* 5, 4 (2025). doi:10.3390/architecture5040103
[54] Johannes Strohm. 2025. Branchen-Bestandsaufnahme zu BIM in der Tages- und Kunstlichtplanung (Stand 06/07 2024). doi:10.5281/zenodo.17169374
[55] Shuaiming Su, Ray Y. Zhong, and Yishuo Jiang. 2025. Digital twin and its applications in the construction industry: A state-of-art systematic review. *Digital Twin* 2, 1 (2025), 2501499. arXiv:https://doi.org/10.12688/digitaltwin.17664.3 doi:10.12688/digitaltwin.17664.3
[56] Fei Tao, He Zhang, Ang Liu, and A.Y.C. Nee. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics* 15, 4 (2019), 2405–2415. doi:10.1109/tii.2018.2873186
[57] Valerian Vanessa Tuhaise, Joseph Handibry Mbatu Tah, and Fonbeyin Henry Abanda. 2023. Technologies for digital twin applications in construction. *Automation in Construction* 152 (2023), 104931. doi:10.1016/j.autcon.2023.104931
[58] Philipp Zech, Peter Burger, Sascha Hammes, David Geisler-Moroder, and Ruth Breu. 2024. BIMReason - Validating BIM Model Correctness. In *Proceedings of BauSim Conference 2024: 10th Conference of IBPSA-Germany and Austria (BauSim Conference, Vol. 10)*. IBPSA-Germany and Austria, Vienna, Austria, 56–63. doi:10.26868/29761662.2024.8
[59] Philipp Zech, Marco Fröhlich, Iolanthe Hochleitner, Sascha Hammes, and Johannes Weninger. 2026. Surrogate Modeling for Efficient Building Assessment and Operation. In *Advances in Information Technology in Civil and Building Engineering*. Springer Nature Switzerland, Cham. accepted for publication.
[60] Philipp Zech, Emanuele Goldin, Christoph Zallinger, Sascha Hammes, Philipp Pobitzer, Judith Michael, and Ruth Breu. 2025. An Ecosystem of DSMLs for Building Commissioning. In *2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 12–23.
[61] Philipp Zech, Sascha Hammes, David Geisler-Moroder, Martin Hauer, Rainer Pfluger, and Ruth Breu. 2025. Quantifying the Simulation-Reality Gap in Building Simulations. In *Proceedings of Building Simulation 2025: 19th Conference of IBPSA (Building Simulation, Vol. 19)*. IBPSA, Brisbane, Australia. doi:10.26868/25222708.2025.1189
[62] Philipp Zech, Sascha Hammes, Emanuele Goldin, David Geisler-Moroder, Ruth Breu, and Rainer Pfluger. 2025. From BIM to Digital Twin: A transformation process through advanced control modeling and automated commissioning using daylight and artificial lighting as examples. *Energy and Buildings* 329 (2025), 115184. doi:10.1016/j.enbuild.2024.115184
[63] Philipp Zech, Alexandra Jäger, Georg Fröch, Rainer Pfluger, and Ruth Breu. 2025. Agile, continuous building energy modeling and simulation. *SIMULATION* 101, 3 (2025), 241–265. arXiv:https://doi.org/10.1177/00375497241251852 doi:10.1177/00375497241251852
[64] Philipp Zech, Christopher Kelter, Sascha Hammes, Judith Michael, and Ruth Breu. 2026. Meta-Metamodel-Independent Model Transformations. In *Software Engineering and Advanced Applications*, Davide Taibi and Darja Smite (Eds.). Springer Nature Switzerland, Cham, 329–348.