CIM Wizard: A Flexible Framework for Automated CityJSON Generation from Sparse Urban Data

CIM Wizard: A Flexible Framework for Automated CityJSON Generation from Sparse Urban Data

1st Ali Taherdoustmohammadi
Politecnico di Torino
Turin, Italy
ali.taherdoustmohammadi@polito.it

2nd Daniele Salvatore Schiera
Politecnico di Torino
Turin, Italy
daniele.schiera@polito.it

3rd Edoardo Patti
Politecnico di Torino
Turin, Italy
edoardo.patti@polito.it

4th Lorenzo Bottaccioli
Politecnico di Torino
Turin, Italy
lorenzo.bottaccioli@polito.it

5th Pietro Rando Mazzarino
Politecnico di Torino
Turin, Italy
pietro.randomazzarino@polito.it

***Abstract*—Urban Energy Modelling (UEM) bottom-up approaches involve three phases: *data modelling*, *model extraction*, and *simulation*. The tight coupling of these phases in existing tools limits flexibility and reuse. CIM Wizard addresses the data modelling phase through a pipeline of independent, interchangeable calculators that transform sparse urban datasets into standardised City Information Models (CIMs), while deliberately leaving model extraction and simulation to interoperable downstream tools. Users add new enrichment methods by defining a calculator class and a single JSON configuration entry, with no framework-level changes required. A priority-based fallback mechanism handles missing data, while a topological dependency resolver guarantees correct execution order. Resulting CIMs are stored in a PostGIS spatial database and exported as CityJSON extended with Energy and Utility Application Domain Extensions (ADEs). A real-world Italian case study demonstrates that fragmented municipal data can be transformed into detailed CIMs suitable for urban planning, scenario analysis, and digital twin applications.**

***Index Terms*—Building energy modelling, City information model, Digital twin, Urban data automation, Urban energy modelling**

I. Introduction

Urban Energy Modelling (UEM) is essential for planning sustainable interventions and supporting climate mitigation [1]–[3]. UEM frameworks fall into two main categories: top-down and bottom-up approaches [2]. Top-down methods use aggregated socio-economic data to estimate urban energy demand, typically lacking spatial resolution at the building scale. Conversely, bottom-up approaches model energy use from individual building data, offering high-resolution, scenario-specific analyses [4]. However, their effectiveness is constrained by inconsistent data quality, fragmented sources [5], and the lack of common standards [6].

The general bottom-up UEM workflow (Figure 1) consists of three sequential phases: (i) *data modelling*, in which heterogeneous urban data sources are collected, integrated, and structured into a coherent city information model; (ii) *model extraction*, in which parameterized building and district energy models are derived from that structured representation; and (iii) *simulation*, in which those models are executed to generate energy demand estimates and scenario analyses.

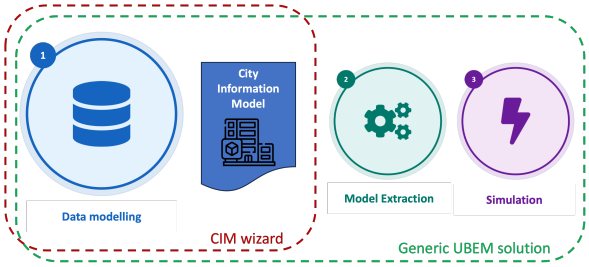

Fig. 1. Conceptual steps of bottom-up UEM

Overcoming the limitations of current bottom-up approaches requires a modular ecosystem in which each phase is addressed independently: (i) frameworks to generate executable parameterized models [7], [8]; (ii) modular co-simulation platforms [9]–[12] to configure and run simulation scenarios; and (iii) standardised City Information Models (CIMs) as a shared, phase-agnostic representation of the case study. City Geography Markup Language (CityGML) [13] with its Application Domain Extensions (ADEs) [14] and CityJSON [15] are the primary standardization candidates for this third role; open-source tools producing compliant outputs across the full data-preparation pipeline remain scarce, while the 3D City Database (3DCityDB) [16] mainly addresses persistence. Standardised CIMs are essential for urban digital twins [17], where interoperable data architectures remain a barrier [18].

Most existing UEM frameworks, such as City Building Energy Saver (CityBES) [19], UBEM.io [20], City Energy Analyst (CEA) [21], SimStadt [22], and UrbanOpt [23], are designed as all-in-one platforms that guide users through the entire pipeline. These frameworks differ in focus and intended application, making their use highly context-specific; they typically employ custom data models tightly coupled to specific simulation engines, restricting interoperability and data reuse across platforms. CIM Wizard addresses this

by focusing exclusively on the first phase, data modelling, and deliberately decoupling it from model extraction and simulation. Its output is a CityJSON-based CIM enriched with Energy and Utility ADEs, providing a standardised, simulation-agnostic representation of the urban case study that most downstream tools can consume.

The limitations identified above translate into four concrete design requirements for a data modelling framework. (1) **Flexible inputs**: the framework must handle heterogeneous, sparse, and incomplete data sources without assuming uniform coverage [24], [25], since real-world urban datasets rarely conform to a single schema or quality standard. (2) **Standardised output**: the framework must produce outputs conforming to an established CIM standard, ensuring reusability across downstream tools and research contexts [26]–[28]. (3) **Application Independence**: the CIM output must be decoupled from any specific simulation engine or model so that the same dataset can be consumed by different downstream frameworks without reformatting [29], [30]. (4) **User extensibility**: users must be able to define, override, or compose feature computation methods with dependency-resolved execution order.

Table I summarizes how CIM Wizard and existing platforms address each of these four requirements.

TABLE I
COMPARISON OF UEM FRAMEWORKS ACROSS THE FOUR IDENTIFIED GAPS. ✓: FULL SUPPORT; ∼: PARTIAL SUPPORT; ×: NOT SUPPORTED.

Framework	Flexible Inputs	Standard Output	Application Independence	User-extensible
CityBES	×	×	×	×
UBEM.io	∼	×	×	×
CEA	∼	×	×	∼
SimStadt	×	CityGML	×	×
UrbanOpt	∼	×	×	∼
CIM Wizard	✓	CityJSON	✓	✓

Unlike all-in-one platforms that tie data preparation to a single simulator, CIM Wizard models Turin once in CityJSON (Fig. 1) and reuses it for external Modelica and `pandapower` runs without re-enrichment (Fig. 4).

Building on our previous work [31], we present **CIM Wizard**, a service-based framework that directly addresses all four requirements. It generates CityJSON-compliant CIMs from sparse, heterogeneous urban data through HTTP/REST interfaces, supports runtime feature computation method selection with automatic dependency resolution, and integrates both public and private data sources, including Open Street Map (OSM), Digital Surface and Terrain Models (DSM/DTM), cadastral databases, and utility networks. The remainder of this paper is organized as follows. Section II describes the system architecture. Section III presents the methodology and default workflow. Sections IV and V present a demonstrative case study and its results. Section VI concludes the paper.

II. Framework Architecture and Components

CIM Wizard is a tool designed to create comprehensive City Information Models from sparse data sources. Throughout this work, we refer to individual data elements (such as building height, construction year, or census population) as *features* of the case study or its components. A key characteristic of CIM Wizard is its extensibility and modular design, which enables the straightforward addition of new features to address diverse requirements without modifying the core engine.

CIM Wizard follows a microservice architecture [32] organized into four layers (Figure 2): the *Data Source Layer* managing incoming data; the *Core CIM Wizard* layer handling orchestration and configuration; the *Representation Layer* ensuring standard representation for the information models; and the *Feature Calculator Modules* performing data enrichment.

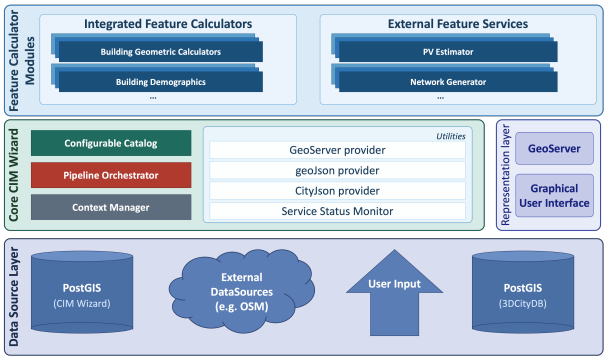

Fig. 2. High-level architecture schema

A. Data Source Layer

The layer comprises four components. *PostGIS (CIM Wizard)* is the central database storing spatial geometries and semantic attributes. Its normalised schema separates geometric entities from descriptive data, which is organised into modular feature sets and indexed by project and scenario identifiers (Figure 3). The same geometry can carry different features across scenarios, supporting efficient scenario switching with incremental updates.

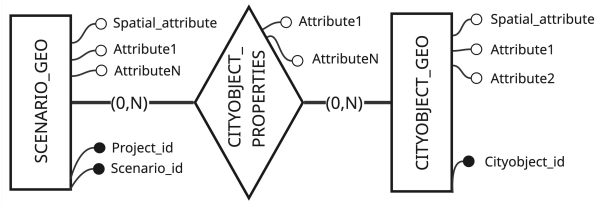

Fig. 3. Generic entity-relations diagram CIM Wizard PostGIS database schema

External Data Sources initialise projects when no user input is available: OpenStreetMap (OSM, via `osmnx`/Overpass API) for global coverage; regional building footprints (Web Feature Service, WFS), census APIs, digital surface models (DSMs), and Typology Approach for Building Stock Energy

Assessment (TABULA) archetypes [**?**] for local context. The *User Input* component accepts boundary polygons, building footprints, or attribute tables in GeoJSON or tabular formats, with configurable schema mapping and fallback priority overrides. *PostGIS (3DCityDB)* provides long-term storage and export, extended with Energy ADE and Utility Network ADE.

B. CIM Wizard Core Layer

The *Pipeline Orchestrator* dynamically constructs and executes processing workflows as described in Section III, invoking feature calculators, handling dependencies, implementing fallback strategies, and managing execution order. The *Context Manager* abstracts all database interactions, enforces schema consistency, and supports automatic migrations. The *Configuration Catalog* stores microservice endpoints, data source locations, feature calculator configurations, method prioritization, and the CityGML mapping schema. The *Utilities* block provides consistency checks, a service status monitor, and format converters (GeoJSON, GeoServer, CityJSON providers).

For scalability, parallel calculators, batched raster queries, and incremental PostGIS updates yield approximately linear cost per pipeline pass; external geospatial APIs are the main bottleneck beyond the 552-building case study.

C. Feature Calculator Modules Layer

Integrated Feature Calculators are embedded Python classes invoked by the orchestrator. Twelve default calculators cover the full building modelling chain from geometry to energy systems. Key default calculators include:

The *Scenario Geometry Calculator* defines the spatial context from a user-provided boundary or derives it via convex hull from building footprints. The *Building Geometry Calculator* validates or acquires footprints from OSM, applies intelligent use-type classification preserving mixed-use patterns, and filters by configurable geometric criteria. The *Building Height Calculator* uses a three-tiered fallback: (1) DSM/DTM difference via batched raster queries, (2) OSM tag extraction with regex parsing, (3) a default 12 m assumption (four storeys at 3.0 m/floor, used only when no elevation signal is available). The *Building Area Calculator* projects geometries to the local Coordinates Reference System (CRS) for accurate computation. *Volume* and *number of floors* are derived from height and area using a configurable storey-height prior (default 3.0 m/floor, tuned in validation when local typology is known). The *Scenario Census Boundary Calculator* retrieves intersecting census zones with demographic variables. The *Building Usage Type Calculator* classifies buildings via OSM tags and geometric filters. The *Building Population Calculator* allocates population proportionally to volume; family counts are inferred from average family size. The *Building Demographic Calculator* combines the above steps into a comprehensive pipeline, including construction-year assignment via weighted distributions derived from census periods. The *Building Geo LoD 1.2 Calculator* constructs semantic surfaces (roof, wall, ground) by extruding 2D footprints, enriched with orientation, area, and azimuth. The *TABULA Building Type Calculator* classifies buildings into TABULA [**?**] archetypes providing U-values, R-values, and materials. The *Thermal Boundary Calculator* applies these to surfaces per the Energy ADE construction module. The *Energy System Calculator* assigns heating and renewable systems (boilers, heat pumps, photovoltaic systems) based on TABULA archetypes with attributes such as fuel type, efficiency, and capacity. Beyond embedded calculators, CIM Wizard supports *External Feature Services* for specialized computations: (i) a *Network Generator and Integrator* for distribution networks in pandapower/GeoJSON format, capable of generating synthetic district heating networks [33]; (ii) a *PV Estimator and Integrator* based on [34] for roof-surface PV estimation.

Finally, the *Representation Layer* comprises *GeoServer* for serving geospatial data over Open Geospatial Consortium (OGC) protocols (Web Map Service, WMS; Web Feature Service, WFS) and a React-based graphical user interface (GUI) for end-user interaction.

III. Framework Methodology and Workflow

CIM Wizard's architecture centers on reusable *features* (e.g., building height, census population), each associated with a *Feature Calculator* module capable of computing the feature via multiple methods. As mentioned above a central orchestrator and configuration catalog enable seamless module integration and plug-and-play customization.

A. Pipeline Orchestration and Execution

A pipeline sequences feature calculators by dependency; independent tasks run in parallel. Fig. 4 shows orchestration, fallback chains (green dashed), and explicit `feature.method | ...` chaining. On failure, lower-priority methods run (e.g., height: DSM, OSM, 12 m default). Runtime API overrides enable calibration without editing configuration files.

B. Context-Aware Data Management

The context manager dynamically manages inputs, dependencies, and intermediate results as a shared memory space. Each calculator retrieves inputs and stores outputs without coupling to other modules. If a required feature has not been computed, the orchestrator automatically resolves upstream dependencies. This enables runtime introspection, debugging, and custom value injection. In addition, the framework accepts both richly detailed and minimally specified user inputs. Users can upload building footprints, census data, or utility layouts in standard formats (GeoJSON, shapefiles, `pandapower`, comma-separated values, CSV). When only a boundary polygon is available, the system queries OSM for building geometries, census APIs for demographics, and elevation services for heights. Missing fields are populated using national defaults or TABULA [**?**] archetypes.

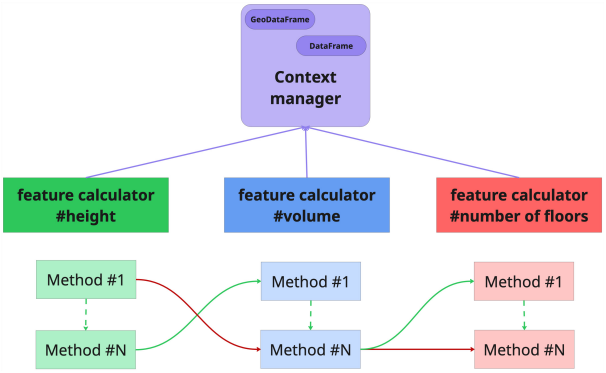

Fig. 4. Pipeline orchestration: user-defined execution paths (red/green), priority-based fallback chains (green dashed), and method chaining within each feature calculator. Outputs are written to the Context Manager.

C. Adding New Feature Calculators

Users extend CIM Wizard by registering a Python calculator class in the configuration catalog (methods, dependencies, priorities); it loads at runtime without orchestration-engine changes.

IV. Case Study Application

To validate the usage of CIM Wizard, we present an end-to-end case study using an external co-simulation platform and external parametrizable models (Figure 1, phases (ii)–(iii)). CIM Wizard's role is strictly upstream (phase (i)): it generates the CityJSON file that parametrizes all building models; the simulation itself is handled entirely by the co-simulation platform described in [11]. The study area, shown in Figure 5, is a neighbourhood in Turin, Northern Italy, comprising 552 buildings (453 residential). Data sources include building footprints from *Geoportale Piemonte* [35], a DSM, and Italian Institute of statistics (ISTAT) census data [36]. A power grid network was also linked to the buildings.

Fig. 5. QGIS visualization of the generated output of CIM Wizard for the case study

Using CIM Wizard's default configuration, we generated a CIM including buildings, heating systems, and a power network, then developed an energy scenario via the co-simulation platform. Six co-simulation components were employed: a *weather file reader* providing a typical meteorological year (TMY) for Turin; a *scheduler* for indoor setpoints (20° C during the heating season); *occupancy readers* for presence and internal gains; *envelope* simulators as Functional Mock-up Unit (FMU)-wrapped Modelica models parameterized from each building's CIM attributes; *heat pump* simulators sized from building area and energy class; and a *power grid model* using `pandapower` for electrical power flow.

Weather drives envelope and heat-pump models; thermal demand and electrical load are exchanged each timestep and forwarded to the grid. Uniform weather, occupancy, and a 10° C setpoint isolate the CIM's data-modelling impact over 15 days in early January (10-min resolution).

V. Experimental Results

We first evaluate CIM Wizard features (Fig. 6–7); Figs. 8–10 illustrate downstream use of the exported CityJSON only.

A. CIM feature validation and sensitivity

Three features—height, floor count, and construction year—were validated against ground truth. Fig. 6(a) shows DSM height bias near +1 m with limited spread, adequate for urban-scale thermal modelling.

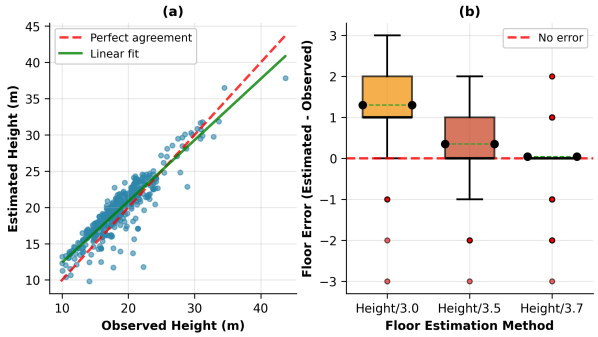

Fig. 6. CIM feature validation and sensitivity. (a) Height error distribution. (b) Floor-count error for alternative storey-height assumptions (sensitivity analysis).

Fig. 6(b) shows sensitivity of floor-count estimates to the storey-height hyperparameter: tuning from the default 3.0 m/floor to 3.7 m/floor for local apartment blocks significantly reduces bias, yielding near-zero median and mean errors; outliers correspond to atypical floor heights. Construction year was validated at ISTAT census-period level (Fig. 7), confirming that the modeled age distribution matches the building stock.

Errors in Fig. 6 propagate to volume, TABULA parameters, and the load spread in Fig. 8 even under uniform boundaries.

B. Downstream co-simulation illustration

Fig. 8 shows load curves from external simulators fed by the CityJSON export; variability under uniform boundaries reflects CIM-derived volume, age, and thermal attributes. Fig. 9 correlates volume with heating demand; newer buildings show

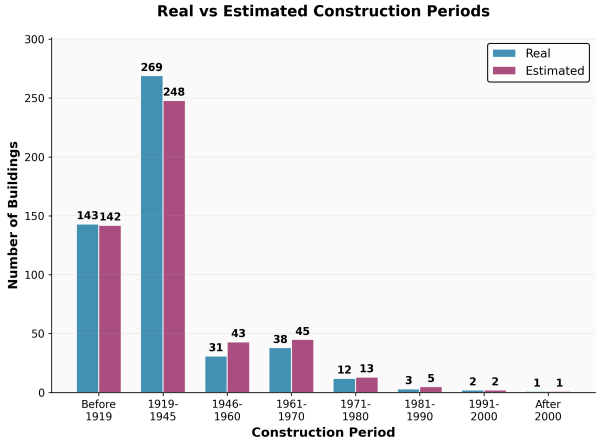

Fig. 7. Estimated vs. ground-truth construction periods (census-period aggregation).

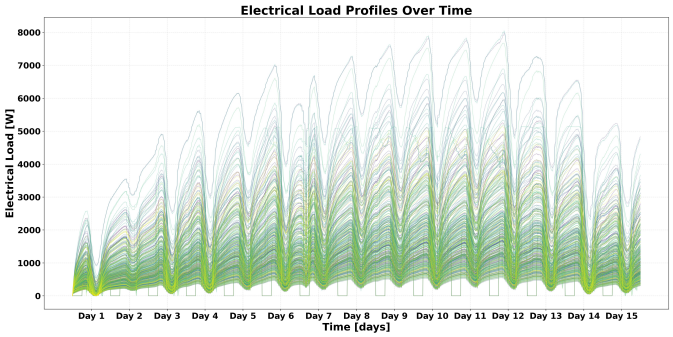

Fig. 8. Electrical load profiles for residential buildings (15-day January window, Turin): external co-simulation fed by the CIM Wizard CityJSON export under uniform boundary conditions.

no efficiency gain, consistent with rapid Italian construction with weak envelope improvements.

The co-simulation with the electrical network (Fig. 10) shows CIM-derived building loads affecting bus voltage magnitudes, highlighting the Utility ADE and grid coupling enabled by the exported CIM.

VI. Conclusion

CIM Wizard is a modular microservice-based framework automating standardised CIM generation from sparse, heterogeneous urban datasets. It operates with user-provided or automatically retrieved inputs, enriching information through customisable fallback strategies, and producing CityJSON outputs extended with Energy and Utility ADEs. By decoupling data modelling from model extraction and simulation (Figure 1), it enables reproducible, interoperable city models consumable by diverse downstream tools. Source code and documentation will be made publicly available upon acceptance. Future work will extend validation to additional cities and features, quantify uncertainty propagation to simulation outputs, and scale orchestration to larger urban inventories.

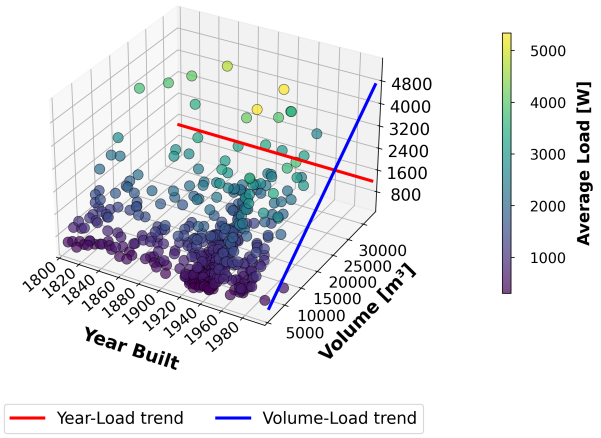

Fig. 9. 3D scatter visualization for average load, volume and building year construction correlation

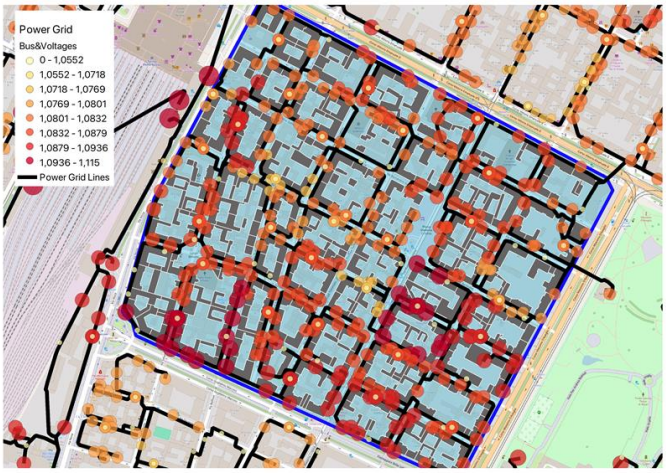

Fig. 10. Snapshot of Simulated Buses' Voltage magnitude in p.u.

References

[1] L. G. Swan and V. I. Ugursal, "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques," *Renewable and sustainable energy reviews*, vol. 13, no. 8, pp. 1819–1835, 2009.

[2] N. Abbasabadi and M. Ashayeri, "Urban energy use modeling methods and tools: A review and an outlook," *Building and Environment*, vol. 161, p. 106270, 2019.

[3] C. F. Reinhart and C. C. Davila, "Urban building energy modeling–a review of a nascent field," *Building and Environment*, vol. 97, pp. 196–202, 2016.

[4] J. Keirstead, M. Jennings, and A. Sivakumar, "A review of urban energy system models: Approaches, challenges and opportunities," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3847–3866, 2012.

[5] Y. Li and H. Feng, "Integrating urban building energy modeling (ubem) and urban-building environmental impact assessment (ub-eia) for sustainable urban development: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 213, p. 115471, 2025.

[6] T. Hong, Y. Chen, X. Luo, N. Luo, and S. H. Lee, "Ten questions on urban building energy modeling," *Building and Environment*, vol. 168, p. 106508, 2020.

[7] P. Remmen, M. Lauster, M. Mans, M. Fuchs, T. Osterhage, and D. Müller, "Teaser: an open tool for urban energy modelling of building stocks," *Journal of Building Performance Simulation*, vol. 11, no. 1, pp. 84–98, 2018.

[8] P. Jayathissa, M. Luzzatto, J. Schmidli, J. Hofer, Z. Nagy, and

A. Schlueter, "Optimising building net energy demand with dynamic bipv shading," *Applied Energy*, vol. 202, pp. 726–735, 2017.

[9] C. Steinbrink, M. Blank-Babazadeh, A. El-Ama, S. Holly, B. Lüers, M. Nebel-Wenner, R. P. Ramírez Acosta, T. Raub, J. S. Schwarz, S. Stark, A. Nieße, and S. Lehnhoff, "Cpes testing with mosaik: Co-simulation planning, execution and analysis," *Applied Sciences*, vol. 9, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/923

[10] T. D. Hardy, B. Palmintier, P. L. Top, D. Krishnamurthy, and J. C. Fuller, "Helics: A co-simulation framework for scalable multi-domain modeling and analysis," *IEEE Access*, vol. 12, pp. 24 325–24 347, 2024.

[11] D. S. Schiera, L. Barbierato, A. Lanzini, R. Borchiellini, E. Pons, E. Bompard, E. Patti, E. Macii, and L. Bottaccioli, "A distributed multimodel platform to cosimulate multienergy systems in smart buildings," *IEEE Transactions on Industry Applications*, vol. 57, no. 5, pp. 4428–4440, 2021.

[12] P. R. Mazzarino, M. Capone, E. Guelpa, L. Bottaccioli, V. Verda, and E. Patti, "A modular co-simulation platform for comparing flexibility solutions in district heating under variable operating conditions," *IEEE Transactions on Sustainable Computing*, vol. 10, no. 2, pp. 408–417, 2025.

[13] Citygml. [Online]. Available: https://www.ogc.org/standard/citygml

[14] O. CityGML. (2023) Citygml ades. [Online]. Available: https://www.citygmlwiki.org/index.php/CityGML-ADEs

[15] Cityjson specifications 1.1.3. [Online]. Available: https://www.cityjson.org/specs/1.1.3

[16] Z. Yao, C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi, and T. H. Kolbe, "3dcitydb-a 3d geodatabase solution for the management, analysis, and visualization of semantic 3d city models based on citygml," *Open Geospatial Data, Software and Standards*, vol. 3, no. 1, pp. 1–26, 2018.

[17] T. Kutzner, K. Chaturvedi, and T. H. Kolbe, "Citygml 3.0: New functions open up new applications," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 88, no. 1, pp. 43–61, 2020.

[18] P. Wate and V. Coors, "3d data models for urban energy simulation," *Energy Procedia*, vol. 78, pp. 3372–3377, 2015.

[19] T. Hong, Y. Chen, S. H. Lee, and M. A. Piette, "Citybes: A web-based platform to support city-scale building energy efficiency," *Urban Computing*, vol. 14, p. 2016, 2016.

[20] Y. Q. Ang, Z. M. Berzolla, S. Letellier-Duchesne, V. Jusiega, and C. Reinhart, "Ubem. io: A web-based framework to rapidly generate urban building energy models for carbon reduction technology pathways," *Sustainable Cities and Society*, vol. 77, p. 103534, 2022.

[21] J. A. Fonseca, T.-A. Nguyen, A. Schlueter, and F. Marechal, "City energy analyst (cea): Integrated framework for analysis and optimization of building energy systems in neighborhoods and city districts," *Energy and Buildings*, vol. 113, pp. 202–226, 2016.

[22] R. Nouvel, K.-H. Brassel, M. Bruse, E. Duminil, V. Coors, U. Eicker, and D. Robinson, "Simstadt, a new workflow-driven urban energy simulation platform for citygml city models," in *Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale*. LESO-PB, EPFL, 2015, pp. 889–894.

[23] R. El Kontar, B. Polly, T. Charan, K. Fleming, N. Moore, N. Long, and D. Goldwasser, "Urbanopt: An open-source software development kit for community and urban district energy modeling," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2020.

[24] O. K. Iseri, A. Duran, I. Canlı, C. M. Akgul, S. Kalkan, and I. G. Dino, "A method for zone-level urban building energy modeling in data-scarce built environments," *Energy and Buildings*, vol. 337, p. 115620, 2025.

[25] D. Perez, "A framework to model and simulate the disaggregated energy flows supplying buildings in urban areas," Ph.D. dissertation, EPFL, 2014.

[26] G. Agugiaro, J. Benner, P. Cipriano, and R. Nouvel, "The energy application domain extension for citygml: enhancing interoperability for urban energy simulations," *Open Geospatial Data, Software and Standards*, vol. 3, pp. 1–30, 2018.

[27] N. Luo, X. Luo, M. Mortezazadeh, M. Albettar, W. Zhang, D. Zhan, L. Wang, and T. Hong, "A data schema for exchanging information between urban building energy models and urban microclimate models in coupled simulations," *Journal of Building Performance Simulation*, vol. 18, no. 3, pp. 333–350, 2025.

[28] F. Rehmann, M. Mosteiro-Romero, C. Miller, and R. Streblow, "Enhancing urban energy modeling: A case study of data acquisition, enrichment, and evaluation in berlin," *Energy and Buildings*, p. 116070, 2025.

[29] M. Issermann, F.-J. Chang, and P.-Y. Kow, "Interactive urban building energy modelling with functional mockup interface of a local residential building stock," *Journal of Cleaner Production*, vol. 289, p. 125683, 2021.

[30] T. Charan, C. Mackey, A. Irani, B. Polly, S. Ray, K. Fleming, R. El Kontar, N. Moore, T. Elgindy, D. Cutler *et al.*, "Integration of open-source urbanopt and dragonfly energy modeling capabilities into practitioner workflows for district-scale planning and design," *Energies*, vol. 14, no. 18, p. 5931, 2021.

[31] P. R. Mazzarino, S. Finocchiaro, L. Barbierato, D. S. Schiera, L. Bottaccioli, and E. Patti, "An automated tool for urban building energy modelling: from sparse datasets to cityjson," in *2024 IEEE International Conference on Environment and Electrical Engineering and 2024 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*. IEEE, 2024, pp. 1–7.

[32] L. De Lauretis, "From monolithic architecture to microservices architecture," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2019, pp. 93–96.

[33] P. R. Mazzarino, M. Capone, E. Guelpa, V. Verda, L. Bottaccioli, and E. Patti, "A multi-agent framework to evaluate energy flexibility in district heating networks," in *2022 IEEE International Conference on Environment and Electrical Engineering and 2022 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, 2022, pp. 1–6.

[34] L. Bottaccioli, E. Patti, E. Macii, and A. Acquaviva, "Gis-based software infrastructure to model pv generation in fine-grained spatio-temporal domain," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2832–2841, 2017.

[35] Regione Piemonte, "Geoportale piemonte," accessed: 2025-08-14. [Online]. Available: https://www.geoportale.piemonte.it/

[36] Istituto Nazionale di Statistica (Istat), "Istatdata — piattaforma per la diffusione dei dati," istatData è la nuova piattaforma; i dati da I.Stat (dati.istat.it) sono stati migrati qui. Accessed: 2025-08-14. [Online]. Available: https://esploradati.istat.it/