MSc thesis in Geomatics

Testing and enhancing the Scenario ADE in the context of building energy performance simulations in Rotterdam

Frederick Auer

June 2026

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

Frederick Auer : *Testing and enhancing the Scenario ADE in the context of building energy performance simulations in Rotterdam* (2026)
This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

The work in this thesis was carried out in the:



3D geoinformation group
Delft University of Technology

Supervisors: Dr. Camilo Léon Sánchez
Dr. Giorgio Agugiaro

Co-reader: Anna Maria Koniari

Abstract

Global urbanisation and climate change mitigation efforts increasingly drive the need for precise urban energy planning and the testing of multiple future scenarios. While Urban Building Energy Modelling (UBEM) and semantic 3D city models, in particular based on the open standard CityGML, provide the foundational framework for these physics-based simulations, they are inherently designed to represent a single, static state of an urban environment. Consequently, existing standards struggle to manage concurrent "what-if" planning scenarios such as building refurbishments or PV device adoption, without duplicating entire datasets. Furthermore, they lack the structured data provenance required to explicitly bind simulation results back to the specific input parameters, boundary conditions, and engine configurations that produced them.

This thesis addresses this data management gap by testing and iteratively enhancing the Scenario Application Domain Extension (ADE) for CityGML 2.0. Utilizing a Design Science Research methodology, the schema was refined from an initial Beta 3 to a fully functional Beta 6 release. The structural requirements for this enhancement were empirically derived by conducting baseline and exploratory SimStadt energy simulations on case-study neighborhoods in Rotterdam. While urban energy served as the empirical proving ground, the schema was deliberately engineered to remain domain-agnostic. Based on these observations and theoretical planning frameworks, scenario data was formally decomposed into three storable pillars: Context, Strategy, and Configuration.

To validate the refined schema, foundational operations and four comprehensive test scenarios —Future Climate, Refurbishment, PV Adoption, and Urban Greening—were encoded into both XML instances and a 3DCityDB relational database. The results demonstrate that the enhanced Beta 6 Scenario ADE successfully tracks object lifecycles, manages parallel scenario branching within a unified "Scenario Multiverse," and reliably links simulation outputs to their exact origins without necessitating the duplication of baseline city geometries. Ultimately, this work provides a robust, domain-agnostic data management layer that supports the structured comparison and reproducibility of multi-temporal urban scenarios, demonstrated here within the context of the energy transition.

Acknowledgements

I will keep this short and sweet… much like the next 165 pages. Thanks to everyone who supported me during this process, especially my supervisors Camilo and Giorgio, with whom many insightful conversations and moments were shared. Throughout the last months, they provided me with great guidance and what felt like unbounded support, always answering my emails and messages almost too quickly. Lastly, I also want to acknowledge the support of all the fellow *geomagicians*, for providing me with free coffee and good company during the long days in the Geolab.

Abbreviations

ADE	Application Domain Extension
BAG	Basisregistratie Adressen en Gebouwen
DDL	Data Definition Language
GML	Geography Markup Language
OGC	Open Geospatial Consortium
SQL	Structured Query Language
UBEM	Urban Building Energy Modelling
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WWR	Window-to-Wall Ratio
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSP	Exploratory Scenario Planning

Contents

1. Introduction 1
1.1. Modelling the Energy Transition with 3D City Models 1
1.2. The Scenario Problem 2
1.3. Problem Statement 2
1.4. Thesis Objective 3
1.5. Research Questions 3
1.6. Reading Guide 3

2. Background and Related Work 5
2.1. Scenarios in Urban Planning 5
2.2. Urban Building Energy Modelling 7
2.3. Semantic 3D City Models 8
2.4. Tracking Change in 3D City Models 10
2.5. The Scenario ADE 13

3. Methodology 15
3.1. Overview 15
3.2. Requirement Evaluation through Conducting Empirical Simulations 16
3.3. Scenario Decomposition Framework 17
3.4. Iterative Schema Refinement 21
3.5. Final Encoding and Validation Approach 23

4. Schema Development and Evaluation 25
4.1. Empirical Analysis of Urban Energy Simulations 25
4.2. Scenario Requirements Development 32
4.3. Scenario ADE Evaluation and Refinement Process 35
4.4. Implementation of Final Encodings 64

5. Results 69
5.1. Energy Simulation Run Results 69
5.2. Scenario ADE Results 75

6. Discussion 123
6.1. Key Findings 123
6.2. Interpretation 123
6.3. Implications 125
6.4. Limitations 127
6.5. Recommendations 128

7. Conclusion 131
7.1. Research Questions 131
7.2. Future Work 134
7.3. Reflection 135

A Declaration of AI/LLM usage 137

B Reproducibility self-assessment 139
B.1 Tools 139

C Appendix 141

C.1 Xlink and Xpaths 141
C.2 Window to Wall Ratio Outlier Detection 143
D Appendix (SQL Queries) 149
Bibliography 153

List of figures

Figure 2.1 The process of classifying a phenomenon as a scenario by Spaniol & Rowland (2019) from a literature review of scenario definitions. 6
Figure 2.2 Overview of UBEM methods (Kong et al., 2023). 7
Figure 2.3 Overview of existing UBEM tools Ferrando & Causone (2019)). 8
Figure 2.4 The Model-Driven Approach (MDA) for developing a CityGML ADE, as illustrated by Bachert et al. (2024) building on the foundational work of Van Den Brink et al. (2013). 10
Figure 2.5 Version and VersionTransition logic used in CityGML 3.0 Chaturvedi et al. (2017). 11
Figure 3.1 The four phases of the thesis methodology. 16
Figure 3.2 SimStadt urban energy simulation reference workflow, adapted from Gao et al. (2025). 17
Figure 3.3 Scenario Pillars: the Context, the Strategy, and the Configuration. 18
Figure 3.4 DSR methodology applied to the Scenario ADE. 21
Figure 3.5 Scenario ADE evaluation and refinement process. 22
Figure 4.1 Case-study neighbourhoods in Rotterdam. 26
Figure 4.2 Input/output view of a standard baseline Energy Demand SimStadt Workflow run for a given 3D city model. 27
Figure 4.3 Input/output view of the three SimStadt workflows used in this thesis, overlaid on the baseline. 28
Figure 4.4 The exploratory simulation runs, Standard Heating Demand Analysis runs varying one input from the baseline, and runs with the PV-potential and refurbishment workflows added. 29
Figure 4.5 The simulation runs carried forward as the four test scenarios used to evaluate the Scenario ADE. 31
Figure 4.6 Package diagram summarising the findings of Phase 1 and the initial understanding of what a Scenario ADE must be able to represent. 32
Figure 4.7 Scenario ADE beta 3 UML diagram 37
Figure 4.8 Scenario ADE beta 3 *AbstractGenericParameter* 38
Figure 4.9 UML of the Core Modules for Scenario ADE beta 5 46
Figure 4.10 UML of the Scenario Module for Scenario ADE beta 5 47
Figure 4.11 CityGML 3.0 Versioning Module 52
Figure 4.12 UML of the Versioning Module for Scenario ADE beta 6 54
Figure 4.13 UML of the Core Module for Scenario ADE beta 6 56
Figure 4.14 UML of the Scenario Module for Scenario ADE beta 6 57
Figure 4.15 UML of the Simulation Library for Scenario ADE beta 6 58
Figure 4.16 Two placements for a result when one building takes part in two scenarios. Left, results on the object do not know their scenario. Right, results on the scenario stay separated and still point back to the building. 59
Figure 4.17 Overview of Feijenoord Scenario Creation 66
Figure 4.18 Scenario Multiverse Tree for Feijenoord 67
Figure 5.1 Absolute Comparison PEREZ vs HAY 70
Figure 5.2 Relative Comparison Perez vs Hay 70
Figure 5.3 Barchart of standardized values to window wall ratio's for Feijenoord 71

Figure 5.4 Barchart of standardized values to window wall ratio's for Prinsenland 72
Figure 5.5 Per-building demand difference distribution for Feijenoord (20% archetype against measured AI). Left: heating. Right: cooling. The dashed lines mark the quartiles and the interquartile-range outlier bounds. 73
Figure 5.6 The most extreme WWR outlier buildings, highlighted in red. Top row, cooling outliers: a Feijenoord residential unit (+1505.3%) and two Prinsenland residential units (+701.4%, +685.5%). Bottom row, heating outliers: a Feijenoord industrial building (-64.6%), a Prinsenland production building (-56.4%), and a Feijenoord production building (+103.5%). Cooling outliers are small low-demand units, while heating outliers are larger non-residential buildings. The full per-building tables and the matching 3D model views are in Section C.2. 74
Figure 5.7 Neighbourhood-total change against the baseline for the Feijenoord what-if runs. Heating and cooling are shown relative to the as-built 2025 baseline, and the PV columns under urban greening are relative to the no-tree run. 75
Figure 5.8 The nine foundational operations used as building blocks for the test scenarios. 77
Figure 5.9 Operation 1 - Inserting a new root element (building) 78
Figure 5.10 Operation 2 - Deleting a root feature (building) 79
Figure 5.11 Operation 3 - Replacing a root feature (tree) 81
Figure 5.12 Operation 4 - Inserting a new child element (solar panel) 82
Figure 5.13 Operation 5 - Deleting a child feature (solar panel) 84
Figure 5.14 Operation 6 - Replacing a child feature (roof) 86
Figure 5.15 Full Importer/Exporter Database Report for Scenario Multiverse 120
Figure C.1 Top heating outlier, AI to 7% (aerial photo and 3D model view). 145
Figure C.2 Feijenoord top-10 heating and cooling outliers. Left: AI to 7 percent window to wall ratio. Right: AI to 20 percent window to wall ratio. 146
Figure C.3 Prinsenland top-10 heating and cooling outliers. Left: AI to 10 percent window to wall ratio. Right: AI to 20 percent window to wall ratio. 146
Figure C.4 Per-building demand difference distributions for the remaining WWR configurations (heating left, cooling right). Top: Feijenoord 7%. Middle: Prinsenland 10%. Bottom: Prinsenland 20%. The long cooling tail seen in Figure 5.5 recurs in every case. 147

List of tables

Table 1 Design Principles per Pillar and Evaluative Questions. 20
Table 2 Parameter families varied in the Phase 1 further analysis. 29
Table 3 Overview of the four test scenarios used to evaluate the Scenario ADE. 33
Table 4 Scenario 1 Mapping: Concrete data parameters to design principles. 34
Table 5 Scenario 2 Mapping: Refurbishment parameters to design principles. 34
Table 6 Scenario 3 Mapping: PV addition parameters to design principles. 35
Table 7 Scenario 4 Mapping: Vegetation parameters to design principles. 35
Table 8 Summary of Beta 3 validation recommendations. 43
Table 9 Summary of Beta 5 validation recommendations. 50
Table 10 Alignment between CityGML 3.0 versioning concepts and Beta 5. 53
Table 11 Summary of Beta 6 validation assessment and forward recommendations. 63
Table 12 Comparison of standardized values to window wall ratio's Feijenoord 71
Table 13 Comparison of standardized values to window wall ratio's Prinsenland 72
Table 14 Reader's guide to the Scenario ADE examples. Each row links an example to what it demonstrates and the design principle (Table 1) it tests. 75
Table 15 File size comparison between storing the four case study scenarios as individual CityGML files versus a single multiverse file. 112
Table 16 Populated Database with all four Scenarios 112
Table 17 Results of the shared engine test. 113
Table 18 Traceable results for a single building in Feijenoord. 113
Table 19 Traceable results for a single surface in Feijenoord 114
Table 20 Distinguishable results for a single building in Feijenoord. 114
Table 21 Distinguishable results for two different simulation engines and settings combinations of the same scenario. 114
Table 22 Recursive version lineage from the scenario chain. 115
Table 23 Branching versions from baseline 115
Table 24 Strategy for the Refurbishment scenario, the `VersionTransition` with its reason, type and the changes it carries. 116
Table 25 Context of the Refurbishment scenario read from the Settings parameters, the weather boundary condition sitting amongst all other inputs. 116
Table 26 Scenario Narrative and Temporal Horizons 117
Table 27 Scenario temporal horizons and Versions 117
Table 28 Parameters per Scenario Simulation Settings, grouped by `scope`. 118
Table 29 `FeatureTransaction` rows, each referencing existing features by id rather than duplicating geometry. 119
Table 30 Prescribed `AttributeTransaction` changes resolved to their exact target feature through `parentFeature`. 119
Table 31 Comparing Cooling Demand of Building between *Refurbishment Scenario 2050* and *Future Climate Scenario 2050* 119
Table 32 Statistics for Feijenoord Neighborhood Scenario Multiverse 120
Table 33 Status of each design principle across the three refined betas. *partial* marks structural support that is not enforced, and *deferred* marks a principle deliberately set aside for a later iteration. 121
Table 34 Tools used in this thesis and their versions. 139

Table 35 Three options for encoding an attribute-level change, shown on a wall window-to-wall ratio set to 0.20. 142
Table 36 WWR Outliers for Feijenoord (7P vs AI) 143
Table 37 Cooling Outliers for Feijenoord (7P vs AI) 143
Table 38 Heating Outliers for Prinsenland (10P vs AI) 143
Table 39 Cooling Outliers for Prinsenland (10P vs AI) 144
Table 40 Heating Outliers for Feijenoord (20P vs AI) 144
Table 41 Cooling Outliers for Feijenoord (20P vs AI) 144
Table 42 AI to 7% Feijenoord Heating Outliers (top 3) 145
Table 43 AI to 7% Feijenoord Cooling Outliers (top 3) 145
Table 44 AI to 20% Feijenoord Heating Outliers (top 3) 145
Table 45 AI to 20% Feijenoord Cooling Outliers (top 3) 146

1. Introduction

The rapid pace of global urbanisation is shaping energy management and climate change mitigation. By 2050, approximately 68% of the world's population is projected to live in cities (UN-Habitat, 2022). This demographic shift drives a large increase in urban energy consumption, where urban areas are already responsible for around 78% of worldwide energy consumption and roughly two-thirds of global greenhouse-gas emissions (Dahlström et al., 2022). Electricity use is now growing twice as fast as energy demand from other sources, with the International Energy Agency projecting that global household electricity consumption will rise by 25% by 2035 and by 60% by 2050, driven in large part by a 70% increase in air-conditioning ownership (International Energy Agency, 2025). In Europe, cooling demand in buildings could increase by more than 100% under future climate scenarios (Dahlström et al., 2022).

These changes are not abstract for the Netherlands. The most recent KNMI'23 climate scenarios, which replace the 2014 set, indicate that Dutch cities will experience stronger warming than rural areas, more frequent tropical nights (minimum temperature above 20 °C) due to the urban heat island effect, and a structural shift from heating-dominated to cooling-dominated demand peaks (Bessembinder et al., 2023).

To respond, the Dutch government has set targets that drive the need for scenario-based energy modelling: a national overarching goal of a 49% reduction in CO_2 emissions by 2030 (Government of the Netherlands, 2019) and, specifically within the housing sector, a 50% reduction by 2050. Achieving this requires the elimination of natural gas for heating in the built environment by 2050 and an ambition to renovate approximately 300,000 existing homes per year (Ebrahimigharehbaghi et al., 2019). At the city level, this transition will require large investments in energy renovation and climate adaptation (FIEC, 2022), and reaching *Positive Energy Districts* (PEDs) requires precise modelling of these interventions to balance supply and demand and avoid grid congestion (Salvalai et al., 2024).

1.1. Modelling the Energy Transition with 3D City Models

To plan and evaluate these interventions, cities increasingly rely on Urban Building Energy Modelling (UBEM). Unlike Building Energy Modelling (BEM), which focuses on individual buildings, UBEM is designed to simulate the energy performance of building stocks at district scale and to account for inter-dependencies (mutual shading, peak load management, shared thermal boundaries) that standalone models cannot capture (Dahlström et al., 2022; Kong et al., 2023; Salvalai et al., 2024).

The simulations are typically run on semantic 3D city models. Their use has been shown to be important for complex large scale energy simulations (León-Sánchez et al., 2025). The most widely used standard in this context is CityGML, issued by the Open Geospatial Consortium (OGC) (Gröger & Plümer, 2012). Unlike purely graphical models, CityGML provides a standardised ontological definition of urban objects, capturing geometry, topology, appearance and semantics (Yao et al., 2018). Domain-specific information that the core model does not cover is added through Application Domain Extensions (ADEs), an established example of which is the Energy ADE (Agugiaro et al., 2018).

1.2. The Scenario Problem

Together, CityGML and the Energy ADE which is needed to further specify the city with additional energy specific characteristics only describe what a city is. They are designed to represent a single, coherent state of an area at a given moment. Urban energy planning, however, does not operate on a single state. To decide whether to refurbish, where to install photovoltaic panels, or which streets to plant with trees, planners need to compare the same neighbourhood under multiple alternative futures. For example, a single neighbourhood under three KNMI'23 climate variants and two refurbishment strategies already produces six alternative city models, each with its own simulation configuration and results. Managing all of these alternatives will be important in supporting decision makers (Zvarikova et al., 2022).

Across the literature this notion of an "alternative future" is captured by the concept of a scenario. The term is used loosely and differently across the different schools of scenario practice. This thesis is informed by the synthesised intuitive logics school's definition of Spaniol & Rowland (2019) and more importantly by the review of urban scenario planning in Abou Jaoude et al. (2022). Both are discussed in Section 2.1. The two together serve as the conceptual reference of what a Scenario is and how they are used.

Once a city is described not by one model but by a set of alternatives, three new questions arise that CityGML does not answer:

1. how to record the lifecycle of a real-world object accross time
2. how to keep parallel alternatives of the same city or neighborhood (refurbishment vs PV vs greening) consistent within one dataset, without duplicating the entire city model for each;
3. how to bind each simulation result to the specific city-model state, simulation engine and parameters that produced it, so that each numerical result can be reproduced and explained.

The three questions reduce to a single underlying need: a way to *track change* in 3D city models that captures not only *what* changed but also *why* and under *which assumptions*. The mechanisms that the CityGML 3.0 versioning module and Dynamizer provide answer the first two of these questions but not the third, as discussed in Section 2.4. The Scenario ADE, which this thesis tests and refines, is intended to close that gap for an existing CityGML 2.0 ecosystem.

1.3. Problem Statement

Despite the capabilities of CityGML and the Energy ADE, the data management of multi-temporal urban energy scenarios still suffers from two structural limitations: No support for scenario analysis and a lack of provenance for results. The first problem has to do with standard CityGML 2.0 models only representing the city at a single point in time. CityGML 3.0 introduces a versioning module which formalises tracking the evolution of a city over time by version transitions (Chaturvedi et al., 2017). However, none of the available mechanisms is designed to manage concurrent hypothetical alternatives, multiple parallel futures of the same neighbourhood within one dataset and within a CityGML 2.0 environment. In practice, citymodel variants are often stored seperately, making comparison across scenarios scattered. The second problem is provenance. Existing extensions, including the Energy ADE, can attach simulated quantities (heating demand, irradiance) to city objects, but they do not bind those values to the specific simulation run, weather file, engine version or parameter set that produced them. Without that binding, simulation results cannot be reliably compared or reproduced.

1.4. Thesis Objective

This thesis tests and further develops the Scenario ADE for CityGML 2.0, in the context of the DigiTwins4PEDs[1] project intentions to optimize urban energy transformation and to generally support the Dutch energy transition, so that it can manage multiple alternative futures of the same neighbourhood within a single dataset and link each future to the simulation that produced it.

The work is scoped to case-study neighbourhoods in Rotterdam. It starts from the existing Scenario ADE at Beta 3 and refines it against requirements derived from baseline and future-climate energy simulations and from the scenario definition adopted in Section 2.1.1. The know-how and insights acquired during the iterative testing carried out during this thesis additionally . The refined conceptual schema (available as XSD file) is validated by encoding a set of foundational operations and four use-case scenarios (future climate, refurbishment, PV adoption, urban greening) into both CityGML XML and a 3DCityDB relational encoding. It is shown to support object lifecycle, parallel alternatives and simulation provenance without duplicating the underlying city model.

1.5. Research Questions

This objective translates into the central question that guides the thesis.

Research Question

How can a Scenario ADE for CityGML 2.0 be developed and implemented to support the structured management, analysis, and comparison of multi-temporal scenarios in urban applications, using urban energy as the case study?

To address this overarching research question, the following sub-questions are formulated:

- *What entities, properties and relations must a Scenario ADE represent to model an urban scenario?*
- *How can an existing Scenario ADE schema be iteratively refined and evaluated against design principles derived from scenario theory and empirical evidence?*
- *To what extent can the operationalised Scenario ADE, in its physical encodings, support the management, analysis and comparison of urban energy scenarios?*

The sub-questions follow a sequential development logic. The first asks what a scenario actually is in structural terms, and turns the theory into the entities, properties and relations the schema must hold. The second takes that structure and iteratively refines an existing schema, evaluating it against the design principles drawn from scenario theory and from the empirical evidence. The third moves the refined schema into its physical encodings and tests how well it can hold and compare real scenarios at neighbourhood scale. Each question builds on the answer to the one before it. Together they answer the main question of how a Scenario ADE for CityGML 2.0 can be developed and implemented.

1.6. Reading Guide

The rest of the thesis is organised into six chapters. Chapter 2 sets out the background concepts, theory and related work the thesis builds on. Chapter 3 lays out the methodology. The schema is refined over repeated iterations following a design science research process, and the refined schema is then transformed into its encodings through a model-driven approach. The tools and

[1] https://www.digitwins4peds.eu/

software used are introduced along the way. Chapter 4 walks through the schema development and evaluation, starting with the analysis of the energy simulations, moving to the scenario requirements, and ending with the iterative refinement of the Scenario ADE and its final encodings. Chapter 5 reports the results. Chapter 6 reflects on what they mean and where the work falls short. Chapter 7 closes with the conclusions and directions for future work.

2. Background and Related Work

This chapter provides theoretical background for the thesis. It first defines what a *scenario* is, drawing on the urban-planning and futures-science debate, and clarifies the distinction between a scenario and a strategy that the rest of the thesis relies on (Section 2.1). It then briefly situates urban building energy modelling (UBEM) and the SimStadt simulator that produces the data the schema must hold (Section 2.2). The chapter then introduces CityGML 2.0 and the Application Domain Extension mechanism on which the Scenario ADE is built (Section 2.3), reviews how the literature has tried to track change, alternatives in 3D city models and provenance (Section 2.4), and finally describes the existing Scenario ADE that this thesis tests and refines (Section 2.5).

2.1. Scenarios in Urban Planning

2.1.1. Defining a Scenario

The starting point for testing and refining the Scenario ADE is a clear understanding of what a scenario actually is.

The concept of a *scenario* was first adopted Herman Kahn at RAND Corporation for military and defence planning and was later developed by Pierre Wack at Royal Dutch Shell in the 1970s, where scenarios helped the company anticipate the 1973 oil crisis (Abou Jaoude et al., 2022). In scenario planning practice literature, three schools are commonly distinguished (Bradfield et al., 2005). The Intuitive Logics school, developed at SRI, Shell and the Global Business Network, is qualitative and narrative and treats plausibility as the measure of rigor (Wack, 1985). The French La Prospective school, formalised by Berger and Godet, combines structured analytical tools with stakeholder consultation and emphasises preferred futures (Godet, 1986). The Probabilistic Modified Trends school is more quantitative and relies on trend and cross impact analysis, treating probability as the measure of rigor. Each school answers the question of what counts as a scenario differently. Strict definitions drawn from one school often exclude scenario forms recognised by the others.

Within urban planning specifically the operational categorisation, reviewed by Abou Jaoude et al. (2022) and based on the tripartite division of Börjeson et al. (2006), distinguishes three planning approaches. These are predictive scenarios (forecasting), normative scenarios (visioning and backcasting), and Exploratory Scenario Planning (XSP). Urban planning practice rarely uses any of the three in isolation and increasingly relies on mixed and hybrid approaches that combine techniques across the categories (Abou Jaoude et al., 2022).

What counts as a scenario across these categories has long been a subject of debate. Khakee (1991) introduced scenarios into urban planning and noted ambiguity surrounding their definition. Bradfield et al. (2005) and others Varum & Melo (2010) describe the field as suffering from a "methodological chaos" of competing definitions.

Spaniol & Rowland (2019) conducted a scoping review of more than 400 published claims of the form "a scenario is…" to bring some order to definitions within the Intuitive Logics tradition, the school from which XSP descends (Abou Jaoude et al., 2022).

Rather than proposing a new definition, the authors synthesised a shared definition from the existing literature. The synthesis identifies six core characteristics organised into three broader categories:

1. **Scenario Purview:** the primary way scholars define a scenario takes place in the future and is shaped by external contextual factors.
2. **Normative Qualities:** a "good" scenario takes the preferred form of an internally consistent, plausible narrative.
3. **Differentiated Set:** scenarios are produced in groups that have limited overlap with one another.

Arranged through the order of prominence in the literature, Spaniol & Rowland (2019) constructed a flowchart that distinguishes scenarios from what aren't. (Figure 2.1).

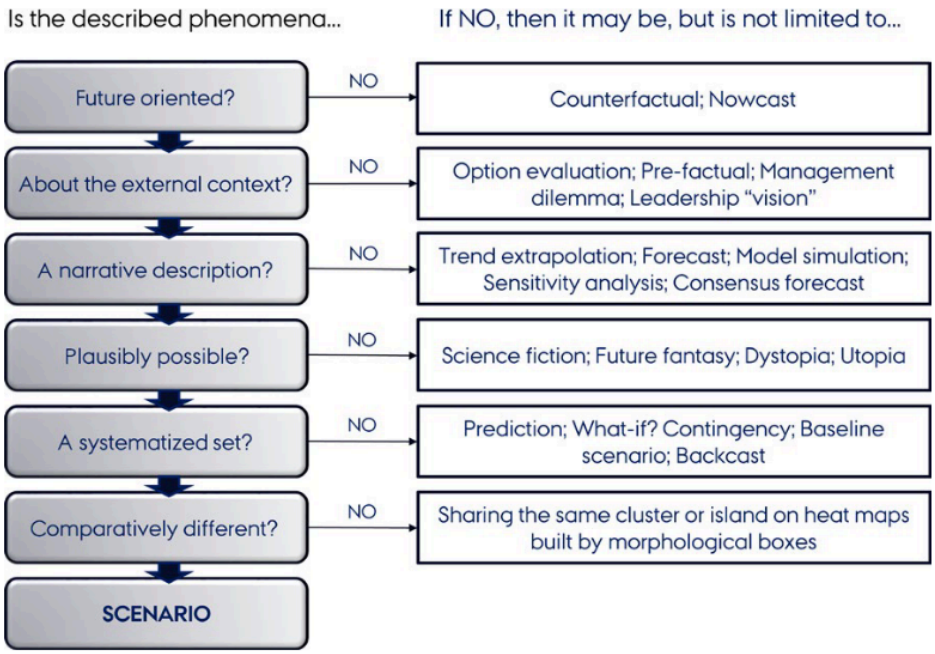


Figure 2.1: The process of classifying a phenomenon as a scenario by Spaniol & Rowland (2019) from a literature review of scenario definitions.

The synthesis is used in this thesis as a conceptual reference for what a scenario can look like. The flowchart classifies predictive forecasts and isolated what-if explorations as non-scenarios, which excludes artifacts that are routinely produced and used across the three urban planning categories above. The thesis therefore treats the synthesis as a reference for general understanding of the scenario definition landscape and to make informed choices moving forward.

2.1.2. Scenario and Strategy

A distinction from the urban planning scenario literature is operationally central to the schema design. Abou Jaoude et al. (2022) states that differentiating between the contextual environment (where planners have no influence) and the transactional environment (where the community is an important actor) is necessary in scenario planning. Without this separation, the assumptions a scenario makes about its surroundings and the actions taken inside those surroundings cannot be told apart. This is why Tapinos (2012) and Moriarty (2012) both emphasise the same separation when evaluating interventions. Therefore, to evaluate interventions, the literature emphasises a clear separation between the future environment and the actions taken within it.

The distinction between the context and an environment where planners can play a role (the strategy) is most explicitly developed in Exploratory Scenario Planning, where it is central to the method (Abou Jaoude et al., 2022). The thesis adopts XSP as the methodological lens through which the schema decomposes scenario work into storable components. The full justification for the decomposition, including how it supports comparison across alternatives and how it is extended with a third Configuration pillar for simulation reproducibility, is given in the methodology (Section 3.3.1).

No evidence has been found that this separation has been carried into an urban modelling schema, let alone an urban energy modelling schema. The Scenario ADE is the artifact through which the thesis will test whether such a separation can be made operational at the schema level.

2.2. Urban Building Energy Modelling

2.2.1. UBEM Approaches

As mentioned in the introduction, one way to test "what-if" scenarios on a city's building stock is through urban building energy modelling (UBEM). UBEM is generally categorised into top-down and bottom-up approaches (Kong et al., 2023). Top-down models allocate aggregated historical energy use to the housing sector using macroeconomic indicators such as GDP, energy prices and population density (Frayssinet et al., 2018), and are primarily used for large-scale assessments and policy forecasting. They cannot model individual buildings or capture the effect of refurbishment strategies because they do not represent the physical properties of the built environment. Bottom-up models, by contrast, calculate energy consumption from the physical characteristics of individual end uses (a single building or a group of buildings) and aggregate the results to the neighbourhood, district or city level (Kong et al., 2023). Within bottom-up models, physics-based (engineering) approaches are the most common in the context of 3D city models (Kong et al., 2023), using building geometry, construction materials, HVAC systems and weather data (Wang et al., 2022). Hybrid approaches that combine physical equations with statistical models for uncertain parameters (e.g., occupant behaviour) are seen as a research opportunity but still face challenges in data access, generalisation and scalability (Abbasabadi & Ashayeri, 2019; Kong et al., 2023). Figure 2.2 gives an overview of UBEM methods, divided into top-down and bottom-up.

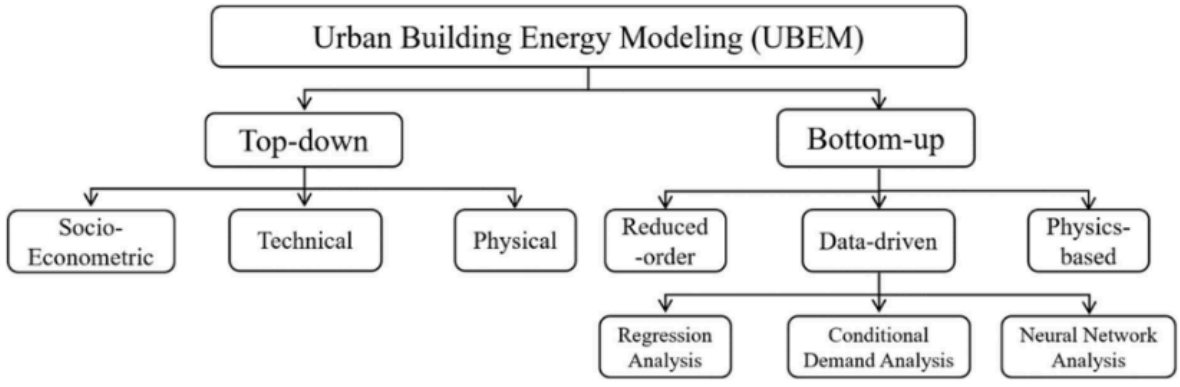


Fig. 1 Schematic diagram of UBEM methods

Figure 2.2: Overview of UBEM methods (Kong et al., 2023).

2.2.2. Physics-based UBEM Tools and SimStadt

There exists many different UBEM tools depending on the context (Figure 2.3). Tools like EnergyPlus focus on detailed single-building physics, while tools like SimStadt (Nouvel et al., 2015) are designed for district scale simulations and customisation (Ferrando & Causone, 2019). SimStadt is the tool used throughout this thesis, because of its use in previous DigiTwin4PEDs research for energy modelling simulations. Therefore, the initial requirements for the Scenario ADE take inspiration from the input/output structure of a SimStadt run, and is characterised in (Section 4.1).

Tool	Website	Developer	Typology	Energy simulation	Presented in
CitySim	https://citysim.epfl.ch/	EPFL	Software in Java and C++	Equivalent electrical circuit	(Robinson et al., 2009)
SimStadt	http://www.simstadt.eu/de/index.jsp	University of Stuttgart	Software	ISO 13790	(Nouvel et al., 2015)
Umi – Urban modelling interface	http://urbanmodellinginterface.ning.com/	Sustainable Design Lab, MIT	Plug-in for Rhinoceros 5	EnergyPlus	(Reinhart et al., 2013)
CityBES – City Building Energy Saver	https://citybes.lbl.gov/	LBNL	Online Platform	EnergyPlus	(Hong et al., 2016)
CEA – City Energy Analyst	https://cityenergyanalyst.com/	ETH Zürich and Singapore	Plug-in for ArcGIS	(Fonseca and Schlueter, 2015)	(Fonseca et al., 2016)
TEASER - Tool for Energy Analysis and Simulation for Efficient Retrofit	https://github.com/RWTH-EBC/TEASER	RWTH Aachen University	Open source on Phyton	Design-driven reduce order model	(Remmen et al., 2018)

Figure 2.3: Overview of existing UBEM tools Ferrando & Causone (2019)).

2.3. Semantic 3D City Models

2.3.1. CityGML 2.0

CityGML (City Geography Markup Language) is an international standard issued by the Open Geospatial Consortium (OGC) and serves as an open data model and XML-based exchange format for storing and exchanging 3D city models. These models are semantic in that they describe the urban environment based on thematic and logical criteria rather than just graphical appearance (Yao et al., 2018). CityGML objects represent real-world entities (buildings, roads, bridges, vegetation, etc.) and store their geometry, topology, appearance and semantics through attributes such as class, function and usage (Gröger & Plümer, 2012). The standard has a modular structure with a core module that defines the basic components and abstract base classes, and a number of thematic modules (Building, Bridge, Transportation, Vegetation, WaterBody, Relief and LandUse) that extend the core.

A distinction must be made between CityGML as a *conceptual data model* and its *physical encoding*. The conceptual model is a standardised ontological framework, typically represented in UML, that specifies how urban space is decomposed into thematic modules and how spatial entities link to the physical and operational attributes required for downstream applications such as energy simulation (León-Sánchez et al., 2025; Wang et al., 2022). To support different workflows, the conceptual model is then realised through one or more physical encodings, where the relationship to the underlying UML is defined through a mapping process. CityGML is traditionally serialised as an application schema of the Geography Markup Language (GML), with an XML-based encoding whose structural constraints are enforced by XML Schema Definition (XSD) files. A more recent alternative is CityJSON, a lightweight JSON-based encoding (Ledoux et al., 2019). CityJSON improves web performance, but it represents a lossy encoding of a subset of the CityGML conceptual model and trades complexity for compactness.

Because CityGML is designed as a general purpose standard, it lacks the specific attributes, features and city objects required for specialised domains such as energy or scenario simulations.

To address this, it includes an extension mechanism called the Application Domain Extension (ADE), introduced in Section 2.3.3. CityJSON offers a conceptually similar mechanism called Extensions, which can also add new attributes, city object types and properties (Ledoux et al., 2019). The CityJSON Extension mechanism is more constrained than an ADE.

2.3.2. Storing CityGML: 3DCityDB and cjdb

Working with large CityGML datasets requires storage outside of single XML files, typically in a database. Two primary solutions exist. The 3D City Database (3DCityDB) is the established open-source solution that maps the CityGML schema onto a relational database management system (RDBMS). The architecture of the 3DCityDB 4.x translates the object-oriented hierarchy of CityGML into a relational (E-R) schema, requiring 66 tables for CityGML 2.0 (Yao et al., 2018). The newly released 3DCityDB 5.0, adds support for CityGML 3.0 and significantly reduces the number of tables by a complete reingeneering of the database schema. However, it still lacks official support for ADEs and therefore remains unused in this thesis.

More recently, Powałka et al. (2024) introduced cjdb, a leaner alternative designed for CityJSON. It uses a feature-centric approach in which the JSONB type stores city objects directly in PostgreSQL. This results in far fewer tables and faster ingestion than 3DCityDB, but cjdb still lacks the maturity to perform spatial queries on the geometry of individual semantic surfaces such as walls or roofs (Powałka et al., 2024).

2.3.3. Application Domain Extensions

CityGML provides two ways to add domain-specific information: *generic attributes* and *Application Domain Extensions*. Generic attributes allow extra data to be attached to a city object, but with no fixed semantics. ADEs, by contrast, provide explicitly modelled feature types with well-defined semantics, attributes and relationships (Biljecki et al., 2018). An ADE is specified in XSD files that refer back to the core CityGML XSDs and uses its own XML namespace alongside the standard CityGML namespaces. The result is much better interoperability between actors and software tools than ad-hoc generic-attribute extensions.

2.3.3.1. Energy ADE

The Energy ADE is the most mature ADE in the context of this thesis. It has gone through several versions. Version 1.0 was released in 2018 (Agugiaro et al., 2018). The latest version, 3.0, is still in active development within the DigiTwins4PEDs project (Agugiaro & Padsala, 2025). None of these versions add native support for scenarios. It extends the core CityGML data model with energy-specific concepts such as thermal zones, construction material properties, occupancy schedules and time-series data, none of which the core model can represent on its own (Agugiaro et al., 2018; Agugiaro & Padsala, 2025). The Energy ADE provides what is needed to define and run a bottom-up physics-based simulation on a CityGML model, but it represents a static snapshot of a city at a single point in time. It has no native support for "what-if" analysis or for managing the results of multiple simulations on the same city. As argued in the introduction and developed in Section 2.5, the Scenario ADE is intended to bridge this gap.

2.3.4. Approaches to Developing an ADE

The development of a CityGML ADE follows a structured methodology that ensures domain-specific knowledge is formally defined and technically interoperable. The standard methodology was established by Van Den Brink et al. (2013) and uses a model-driven approach. Bachert et al. (2024) applies the same approach to map the Energy ADE 1.0 (originally developed only for CityGML 2.0) to CityGML 3.0. The work moves the focus from the structural framework to the

data-integration challenges of mapping building physics and thermal properties into inherited CityGML structures. The MDA workflow used by Bachert et al. (2024), based on Van Den Brink et al. (2013), is summarised in Figure 2.4. It begins with a conceptual UML model and ends with the automatic derivation of the XML schema through ShapeChange software[2].

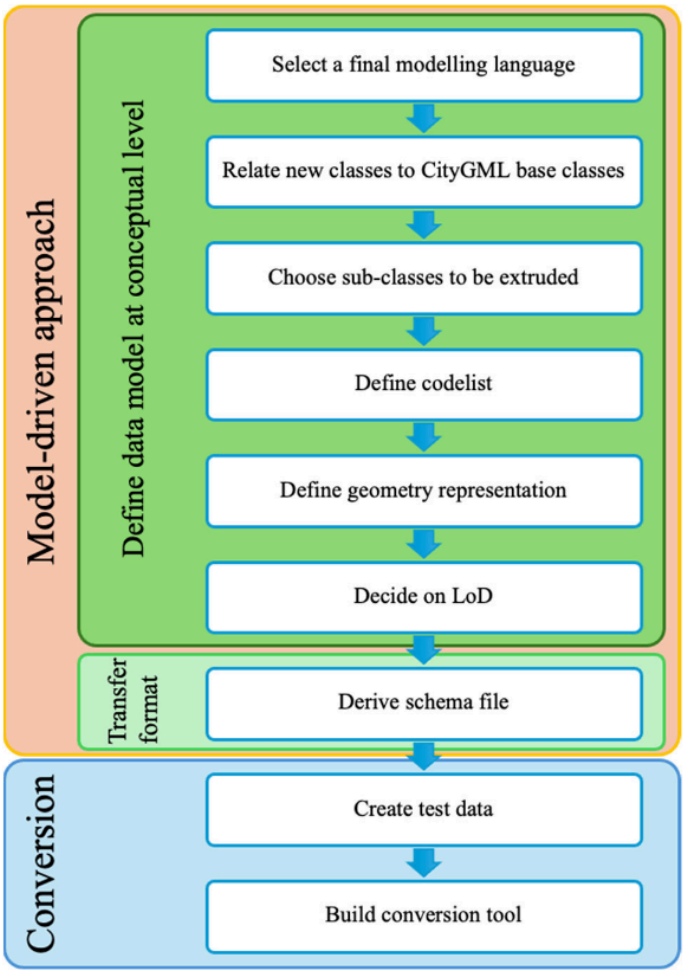


Figure 2.4: The Model-Driven Approach (MDA) for developing a CityGML ADE, as illustrated by Bachert et al. (2024) building on the foundational work of Van Den Brink et al. (2013).

The MDA assumes that an ADE is built from scratch, starting from requirements. When an ADE already exists, as is the case here, the research changes towards evaluation and refinement, and then creation. This change is presented in the methodology chapter (Section 3). The MDA itself is reused in this thesis to translate the refined UML into XML and SQL encodings (Section 3.5).

2.4. Tracking Change in 3D City Models

3D city models are usually treated as static snapshots, yet cities are dynamic and the models that describe them must be too (Chaturvedi et al., 2017; Vitalis et al., 2019). Tracking change in this context has at least three different meanings:

1. recording the lifecycle of a real-world object as it is built, modified or demolished;
2. representing parallel alternatives of the same city (e.g., two different scenarios);

[2] https://shapechange.net/

3. keeping a trace of why a representation changed, which process produced it and how the results of that process can be stored alongside the change itself.

The literature has mostly focused on the first two, with a couple standards addressing provenance. However, achieving all three together in a functional ADE layer, linking the *what changed?* to *which process?* and answering *how?*, *why?* is what the Scenario ADE addresses and the one CityGML's existing mechanisms still leave open.

2.4.1. Versioning in CityGML 3.0

The CityGML 3.0 versioning module, derived from Chaturvedi et al. (2017), introduces versions and version transitions as first class feature types. A `Version` is a named snapshot of a set of features at a given point in time. A `VersionTransition` represents the causal link between two versions and carries the transactions that turn the predecessor into the successor. Transactions are typed as `insert`, `delete` or `replace`, and each refers to a `newFeature`, an `oldFeature`, or both (Chaturvedi et al., 2017). The transition itself is typed as `planned`, `realized`, `historicalSuccession`, `fork` or `merge`, which lets a CityGML document describe both linear historical succession and branching planning alternatives within a single dataset. These transitions take inspiration from git-like versioning systems.

Versions, version transitions and the features they contain are all bitemporal. The lifespan attributes `creationDate` and `terminationDate` record the *database* time when the representation was added or removed, while `validFrom` and `validTo` record the *real-world* time during which the object existed (Chaturvedi et al., 2017). This separation is taken from INSPIRE and lets a user query both "how did the city look at time t" and "how did the database describe the city at time t". To keep object identity stable across versions, an object carries a *major ID* (its `gml:identifier`) that remains the same across versions, and a *minor ID* embedded in its `gml:id` that distinguishes specific versions of the same real-world entity.

Figure 2.5 summarises the resulting structure. It is important for what follows because the Scenario ADE refinement process in this thesis led to a version which adopts much of this vocabulary directly, even though it targets CityGML 2.0 rather than 3.0 (Section 2.5.2).

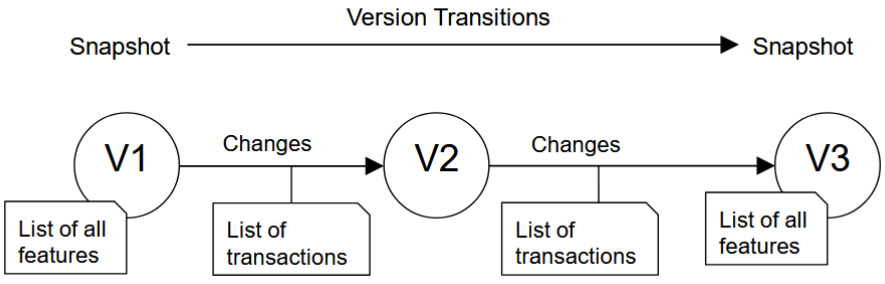


Figure 2.5: Version and VersionTransition logic used in CityGML 3.0 Chaturvedi et al. (2017).

2.4.2. The Dynamizer

The Dynamizer is the second mechanism CityGML 3.0 provides for representing change. Where the versioning module deals with how features come and go, the Dynamizer deals with how a single static attribute of a feature varies over time. A `Dynamizer` carries an `attributeRef` (an XPath-like reference to the attribute it animates), a `startTime` and `endTime`, and a `dynamicData` element that holds the time-varying signal as an `AtomicTimeseries` (a single series)

or a `CompositeTimeseries` (multiple series combined). It can also optionally link to an external sensor through a `SensorConnection` to integrate live IoT data.

2.4.3. Alternative Versioning Approaches

Outside the OGC framework, two alternative approaches should be mentioned. The first is distributed version control systems (DVCS) such as GeoGig, which treat geospatial information as flexible content suitable for non-linear workflows rather than static rows in a table (Wieland et al., 2017). The second is Git-based versioning over CityJSON, which exploits the compactness of the JSON encoding to track changes through a modified directed-acyclic-graph (DAG) structure (Idris et al., 2024; Vitalis et al., 2019). Idris et al. (2024) compares the two encodings directly and concludes that CityJSON's flatter structure makes versioning simpler and storage more compact, while CityGML's hierarchical structure makes versioning more verbose but offers better control over individual semantic surfaces.

These approaches both treat a city model as a tree of files or features and apply a generic logic on top. They are very effective for tracking what changed and when, but, like the CityGML 3.0 versioning module, they do not describe *why* a change was made or which process produced it.

2.4.4. Tracking Provenance

The third part is the keeping track of the provenance of how the data was produced, usually called lineage in the geospatial world. Two standards describe it. The ISO 19115 metadata standard (ISO, 2014) describes a result through its sources and process steps recording the algorithm, software, parameters, and inputs or outputs. As shown by Liping Di et al. (2013), this model can capture the full run of a geoprocessing workflow automatically. Their research includes storing NDVI data with provenance back to its processing workflow. Alternatively, the W3C PROV data model (Belhajjame et al., 2013) is the general, domain agnostic standard for provenance on the web. The PROV data model has three core types (entity, activity, and agent) to link a result to the activity that generated it, the inputs used, and the agent responsible.

These two overlapping models have been aligned to improve interoperability (Jiang et al., 2018). Jiang et al. (2018) tests this alignment on a geospatial use case recording a map passing through versions where new features are added and attributes are resurveyed, each documented with its provenance. The difference is that tracking chronological updates to a single map is not exactly the same as what the Scenario ADE needs.

The ISO 19115 standard has been brought to CityGML directly. Dietze et al. (2007) mapped ISO 19115 lineage classes onto 3D city models at the dataset level. Labetski et al. (2018) took this further into a Metadata ADE that stores ISO 19115 metadata at the dataset, thematic, and individual feature levels. They note that there is confusion between feature level metadata and attributes. The metadata is the descriptive text about how the input features were originally created and not what the attributes describe themselves.

Structured provenance inside CityGML is therefore feasible, but the existing applications describe how input data was made rather than tracking results or changes that vary across scenarios back to the processes that created them. They are additionally quite complex and both standards rely on their own data models.

2.4.5. The Gap This Thesis Addresses

Putting the four mechanisms next to each other:

- CityGML 3.0 versioning manages parallel alternatives and lifecycle at the *feature* level, but does not link a change to the simulation or analysis that produced it.

- The Dynamizer animates a single attribute over time, but cannot represent branching alternatives or hold scenario-specific results.
- Git/GeoGig-based versioning tracks file-level deltas, but is agnostic to the semantic structure of a scenario and to its inputs and outputs.
- Provenance standards and the Metadata ADE track how input data was originally created, but do not bind internal changes and outcome attributes to the specific assumptions and tools that produced them. Let alone for when there are many alternative "what-if" scenarios which structurally also require some form of comparability.

None of the mechanisms link *what* changed to *why* it changed. None of them stores the tool that produced a result of a possible alternative, the boundary condition under which the change occurred, the assumptions taken to produce it, or the result along side the change itself. This is a mechanism of provenance that the Scenario ADE aims to provide.

2.5. The Scenario ADE

2.5.1. Origins

The first version of the Scenario ADE is was developed at AIT (Austrian Institute of Technology) and published in 2017[3]. It establishes a first interpretation where a Scenario as a unique composition of objects, parameters, operations and resources, and the ability to derive one scenario from another. It is a starting point and simple model which still lacks several of the capabilities required by the use cases in this thesis. There is no versioning of features or attributes in the CityGML 3.0 sense, no clean separation between scenario context, intervention and configuration, and no support for hierarchical feature insertion (a new feature added as a child of an existing one). The method in which data is stored is also not yet finalised, with reliance of EnergyADE concepts and no structured way to store or link results, assumptions or provenance to a scenario and the objects it modifies.

Within the DigiTwins4PEDs project further internal iterations of the schema were developed, leading to the Beta 3 version that is the starting artifact of this thesis and will be presented in the implementation chapter (Section 4.3).

2.5.2. Why CityGML 2.0 rather than CityGML 3.0

The CityGML 3.0 versioning module described in Section 2.4.1 provides much of the structural vocabulary that a scenario-aware schema needs. It would therefore be natural to develop the Scenario ADE directly on top of CityGML 3.0. Two reasons motivate the choice to keep targeting CityGML 2.0. First, the surrounding tooling system is not yet mature for CityGML 3.0 and 3DCityDB v5, the relational implementation targeting CityGML 3.0 (Yao et al., 2025), and the broader simulation ecosystem (SimStadt, Energy ADE 3.0) remains anchored on CityGML 2.0. Second, the data and workflows the project depends on, including the datasets used in this thesis, are in CityGML 2.0. Developing the Scenario ADE for CityGML 2.0 therefore keeps it usable within the project's existing ecosystem, while also adding the versioning logic which does not exist on CityGML 2.0

The Scenario ADE evaluated in this thesis takes deliberate inspiration from CityGML 3.0, in particular from the versioning module's `Version`, `VersionTransition` and `Transaction` vocabulary, but adapts it to a CityGML 2.0 environment that does not yet have these constructs natively. Where CityGML 2.0 constraints force a specific implementation pattern, the schema is designed so that pattern can be replaced by its CityGML 3.0 equivalent without redesigning

[3] https://github.com/gioagu/3dcitydb_scenario_ade

the Scenario ADE. Forward compatibility is therefore treated as a design principle in the methodology (Section 3.3.3), and the migration of the Scenario ADE to CityGML 3.0 is left as future work.

2.5.3. What This Thesis Contributes

Building on the existing Scenario ADE described above, this thesis contributes three things. First, it grounds the schema in a formal definition of what a scenario is (Section 2.1.1, Section 2.1.2), which previous Scenario ADE versions did not provide. Second, it evaluates the existing artifact (Beta 3) iteratively against empirical and theoretically derived requirements and refines it through several beta versions into a final Beta 6 (Section 3). Third, it validates the refined schema by encoding test scenarios in both XML and 3DCityDB. (Section 5).

3. Methodology

This chapter explains the four-step methodology used to build and test the Scenario ADE. First, it describes how energy simulations were used to gather initial data requirements. Second, it breaks down the theoretical concept of a scenario into three storable pillars: Context, Strategy, and Configuration to set the design rules for the schema. Third, it outlines the iterative process used to evaluate and improve the schema from early drafts to its final version. Finally, it explains how the final schema is encoded into XML and a 3D database to prove that it actually works.

3.1. Overview

The introduction and the related work identified the gap that motivates this thesis: existing 3D city-model standards can either represent a single city state or version it over time, but none of the available can manage the binding of a change to the assumptions, simulation and parameters that produced it (Section 2.4.5). This is why the Scenario ADE developed in this thesis becomes the artifact that addresses this gap. Because the artifact already exists in an early form (Beta 3, Section 2.5), the methodology of this thesis is not the design of a schema from scratch but the iterative evaluation and refinement of an existing one against requirements derived from both theory and empirical data. It is at the end, once the final version of this thesis Scenario ADE is reached, that the schema is translated into its physical encodings (CityGML XML and 3DCityDB relational schema).

The methodology is therefore structured in four phases, illustrated in Figure 3.1. Phase 1 is conducted to gather the empirical evidence and understanding. A small set of SimStadt simulations characterises an understanding of what is needed for Urban Energy Modeling from which the inputs, parameters and outputs the schema must hold are derived. Some new insights and analysis of parameter sensitivity and using newly acquired data are also conducted and discovered. Phase 2 is conceptual, by drawing on the scenario definition of Section 2.1.1, a scenario is decomposed into pillars from which the design principles for evaluating the schema are derived. Phase 3 is iterative, starting from Beta 3, the schema is evaluated against those principles and four test scenarios, refined and re-evaluated, until reaching the current version, Beta 6. All iterations are created through using UML language. Phase 4 is the testing and validation of the schema where the final Beta 6 is translated into its physical encodings: CityGML XML instances and a 3DCityDB relational schema using the Model-Driven Approach of (Van Den Brink et al., 2013) and (Bachert et al., 2024).

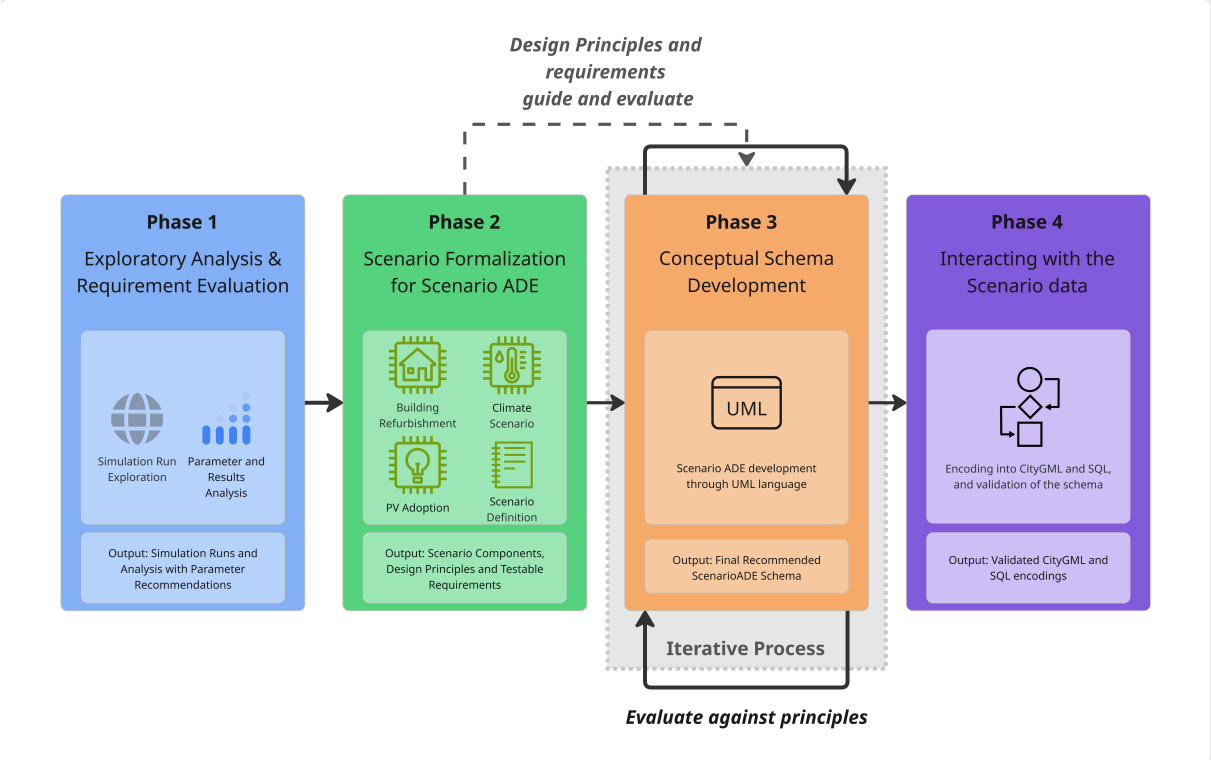


Figure 3.1: The four phases of the thesis methodology.

Each phase is described in detail in the sections that follow.

3.2. Requirement Evaluation through Conducting Empirical Simulations

The development of the Scenario ADE begins with an empirical exploration of physics-based urban energy modelling. Utilizing the SimStadt simulation engine across various neighborhoods in Rotterdam specified by the DigiTwin4PEDs project, multiple simulations are ran to establish a understanding of the inputs, parameters, and outputs.

This phase begins with an analysis of standard SimStadt input, parameters, and output for common energy demand simulations workflows which will be used. Then, different simulations derived from this baseline are ran to explore the parameters, different settings and their respective output results. Further analysis isolating the influence of one parameter at a time (solar radiation model and window-to-wall ratio) is conducted leading to new insights. For example, using the dataset of measured window-to-wall ratios (WWR) from Gemeente Rotterdam as a new input parameter allows for a direct comparison between generalized archetype assumptions and actual per-building characteristics. For this given simulation, the outputs are further analysed through outlier detection. The reference workflow for pre-processing the input 3D city model for these simulations, adapted from Gao et al. (2025), is illustrated in Figure 3.2.

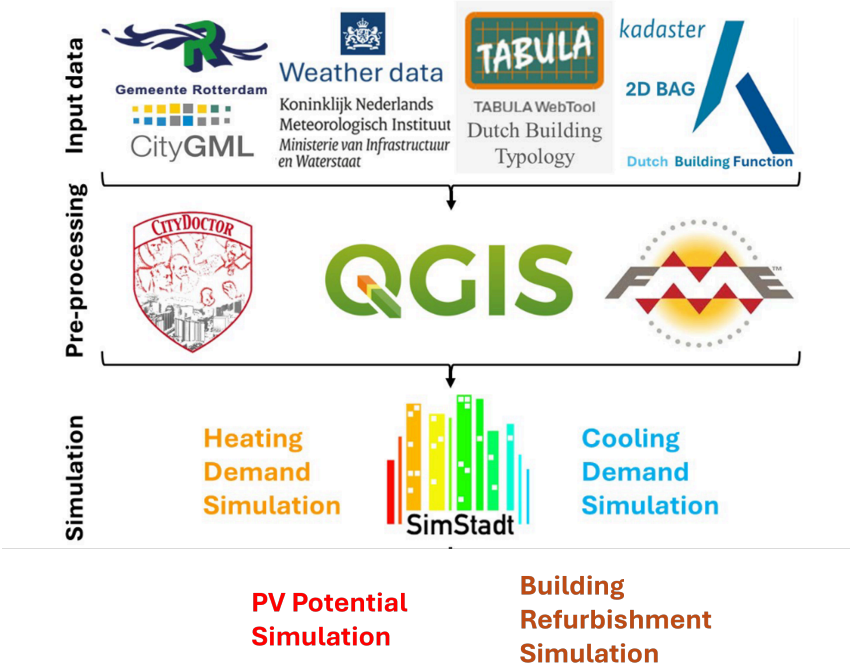


Figure 3.2: SimStadt urban energy simulation reference workflow, adapted from Gao et al. (2025).

The datasets, SimStadt configurations, and parameter values are documented in Section 4.1. The findings of the simulations analysis are reported in Section 5.1. The implications of simulations for the schema design are deduced during the implementation (Section 4.1) and feed directly into developing them into the Scenario Decomposition Framework (Phase 2). These insights are synthesized into a package diagram presented at the end of the implementation, summarizing the data structures required by bottom-up urban energy simulations.

3.3. Scenario Decomposition Framework

The related work introduced a synthesized scenario definition, the three planning approaches that urban scenario practice draws on, and the operational separation between contextual and transactional environments that is central to XSP (Section 2.1). This section operationalises that separation by decomposing scenario work into a set of storable components that the schema can hold.

3.3.1. Decomposing Scenario Work into Storable Components

Translating the theoretical definition of scenarios into a format that can be stored and managed within the Scenario ADE, this research operationalizes a framework motivated by how urban planners approach scenario development (Section 2.1.1). We therefore decompose a scenario into three pillars: *Context, Strategy, and Configuration.*

The *Context* and *Strategy* pillars draw on a fundamental distinction in urban planning literature (Abou Jaoude et al., 2022), separating external environments planners cannot control (Context) from the interventions they can (Strategy). This separation is what makes scenarios comparable. It allows us to hold one constant while varying the others which isolates the effects of specific interventions or conditions. For example, it allows us to test strategies like PV adoption or refurbishment against a fixed context like the KNMI 2050 weather, or vice versa. While this decomposition is most common in Exploratory Scenario Planning (XSP), it serves as a universal container for Predictive and Normative scenarios as well.

The *Configuration* pillar what is added by this thesis to capture the technical reproduceability of the simulation experiments. While the scenario literature mainly addresses definitions rather than the underlying quantitative simulation apparatus, the Configuration was created to explicitly capture the simulation engines, settings, and their output results. This aligns with running urban energy modelling simulations and ensures that all scenarios in this thesis are reproducible and their results fully traceable to their inputs.

Figure 3.3 illustrates this.

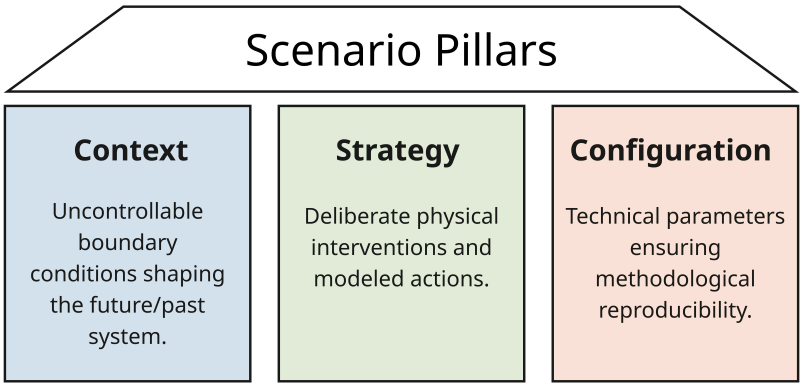


Figure 3.3: Scenario Pillars: the Context, the Strategy, and the Configuration.

1. **The Context (External Context)**

   This pillar holds the external environment and boundary conditions under which the urban system operates. These factors are outside the direct control of the urban planner and form the assumed surroundings of any scenario stored in the schema. Examples include:

   - Climate Conditions: Future weather files derived from standardized climate profiles (e.g., KNMI klimaatscenario's).
   - Socio-Economic Factors: Stochastic variables involving human behavior, such as occupancy schedules.
   - Grid Constraints: Macro-scale limitations related to infrastructure capacities.

2. **The Strategy (Interventions)**

   This pillar holds the physical or technical interventions applied to the buildings or urban area inside a given context. The Strategy and Context together describe what is varied and what is held fixed when scenarios are compared. Examples include:

   - Building Envelope Modifications: Alterations to thermal properties, including U-values, g-values, and wall-to-window ratios (WWR).
   - System Upgrades: Integration of renewable energy sources, such as Photovoltaic (PV) panels.
   - Morphological Changes: Structural transformations including building demolition, reconstruction, or vertical extensions (optopping).

3. **The Configuration (Technical Settings)**

   The configuration pillar ensures the technical reproducibility of the simulation experiment. It should document the technical settings of the simulation environment itself, validating

the qualities and assumptions used for the experiment and providing important metadata tracking. Examples include:

- Simulation Engine: Documentation of the tool utilized (e.g., SimStadt, EnergyPlus).
- Settings: Parameters used to configure the simulation, such as the solar radiation model (e.g., Perez or Hay) or the temporal resolution of the simulation.
- Results: The quantitative outputs produced by the engine under those settings. They are directly linked to the simulation and settings that produced them.

3.3.2. Test Scenarios as Evaluation Cases

Alongside the universal design principles, the Scenario ADE is evaluated against a set of four concrete test scenarios, allowing the schema to be empirically tested. These scenarios are *Future Climate, Refurbishment, PV Adoption, and Urban Greening* and were obtained by dissecting the Phase 1 simulation runs into the three scenario pillars (Context, Strategy, and Configuration). Because the dissection and the resulting concrete parameter values are themselves a Phase 1 outcome, the full scenario definitions are presented in the Implementation chapter (Section 4.2) rather than here.

Methodologically, the four scenarios were also chosen and dissected in order to evaluate the how Context, Strategy, and Configuration dimensions can have a place in a schema. The Scenario ADE is not limited to accommodating these four scenarios however because its initial development stems from energy simulations they are the ground on which the development begins. The thesis follows a non-linear, iterative process in which capabilities discovered during development led to working on a version which can accommodate cases beyond the original scope and not only the energy domain.

3.3.3. Designing Principles as a Guiding Framework

The evaluation of the Scenario ADE was structured through a set of universal schema requirements, derived from two sources and the gathered empirical evidence. The first source is the three pillar decomposition (Section 3.3.1), which is grounded in the XSP context and strategy distinction (Abou Jaoude et al., 2022) and informed by what a scenario is according to the intuitive logics tradition synthesis of (Spaniol & Rowland, 2019). Lastly, the practical schema requirements observed empirically during the Phase 1 simulation runs also play a important role (derived in Section 4.1.5). They are presented here as the evaluation framework that is applied throughout the thesis. The requirements were derived to help support structured reasoning about what a data model must be able to represent. The principles followed in the design of the Scenario ADE are organized into four categories starting with the foundational intentions and followed by the design principles per pillar, summarized in Table 1.

Foundational Intentions

Before the three pillars, there are some architectural intentions that shape the schema as a whole. These are not pillar specific and apply throughout the design process. The first of them, domain agnosticism, did not precede the work but emerged from it. The fuller reasoning is developed in the implementation (Section 4.3). The intentions are as follows:

- **F1 - Domain Agnosticism:** The Scenario ADE should function as a scenario management layer over any CityGML module, or future domain extensions. It should therefore not require further modification. This means the schema must not import or depend on the semantics of any specific domain ADE (e.g., Energy ADE).

- **F2 - Scenario Set Coherence**: A scenario becomes valuable by contrast with alternatives. Both the IL tradition synthesis of Spaniol & Rowland (2019) (criteria on systematised sets and meaningful difference) and the XSP literature in urban planning (Abou Jaoude et al., 2022) treat comparison as central to scenario practice. The schema should therefore support sets of related Scenarios that share a common base state, not only isolated ones.
- **F3 - Forward Compatibility** Design choices should not foreclose migration toward CityGML 3.0. Where CityGML 2.0 constraints require a specific implementation pattern, the schema should be structured so that pattern can be replaced by a CityGML 3.0 equivalent without requiring a complete redesign of the Scenario ADE.

Design Principles per Pillar

The eight principles below are organised by pillar. For each, a short design question is provided to guide design decisions and evaluate the schema.

ID	Principle	Design Question
Context: External Boundary Conditions		
C1	Boundary conditions and interventions remain independently retrievable.	If I removed all intervention data, would the context still make sense on its own and vice versa?
C2	Temporal horizons and narratives travel with the parameters.	Could someone reconstruct the scenario's meaning and timing without reading the raw parameter values?
C3	Parameters carry metadata to identify their role (input vs. result).	If parameter names were anonymized, could we still tell the difference between a setting and a result?
Strategy: Interventions		
ST1	Physical interventions are represented without duplicating base-state geometry.	Does this operation require me to re-encode anything that already exists in the base model?
ST2	Attribute-level changes point unambiguously to their target.	Could an external tool find the exact attribute targeted using only the schema instance?
Configuration: Simulation and Reproduceability Metadata		
CF1	Simulation settings are separable, reusable entities.	If I ran the same setup across ten scenarios, would the configuration need to be defined ten times?
CF2	Results are traceable to the specific object and simulation run.	Given a result value, can I unambiguously identify which building it belongs to and which run produced it?
CF3	Multiple result types for the same object are structurally distinguishable.	If a building has multiple results, is their difference structural or simply a naming convention?

Table 1: Design Principles per Pillar and Evaluative Questions.

Note: Domain Agnosticism

The initial use case for the Scenario ADE was managing energy simulation scenarios, and every case study in this thesis still derives from energy simulations. The intention is presented as a foundational one here so that all schema versions can be evaluated against it, even though the concrete requirements all stem from the Phase 1 energy runs. Where domain-specific examples appear, they are illustrative and show that it can still be used within a specific domain and do not make the Scenario ADE depend on that domain.

Lastly, to ensure also some rigid requirements are met and specific results from the simulations ran in part 1 can be stored, concrete requirements were also derived for each of the four case study scenarios. These are presented in the implementation chapter (Section 4.2.1) and used to evaluate the Scenario ADE versions.

3.4. Iterative Schema Refinement

The evaluation and refinement of the Scenario ADE begins with a pre-existing version, the beta 3 (the artifact). This benchmark begins the iterative methodology of evaluating and refining inspired by design science research (Hevner et al., 2004; Peffers et al., 2007). This process is illustrated in Figure 3.4.

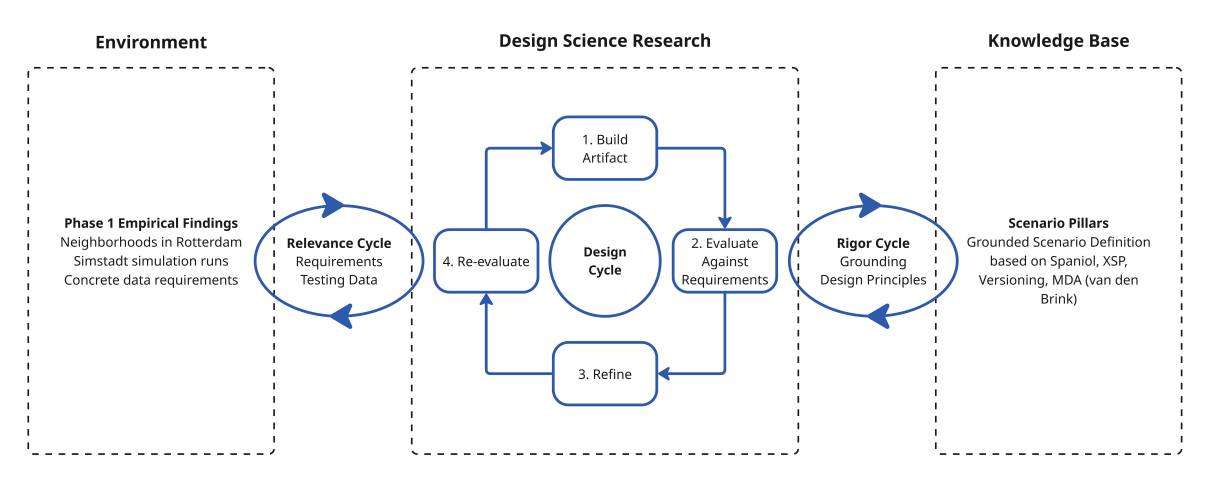


Figure 3.4: DSR methodology applied to the Scenario ADE.

3.4.1. Refinement Strategy Toolkit

The Design Cycle is where the iteration of the Scenario ADE happens. It follows a toolkit of methods, inspired from some of the steps taken during the development of the FloodADE (Luo et al., 2025) which motivates the need to include empirical domain specific requirements, and the QualityADE (Betz et al., 2025) which motivates including some form of validation assessment. These steps also align well with integrating the respective Relevance and Rigor Cycle, inherent to the DSR methodology seen above.

We begin with comparing the artifact (beta 3) to the guiding requirements (Phase 2), using the case study scenarios as illustrative examples. The schema is assessed on the scenario dependent requirements through semantic mapping, and validations assessment. At same cases small encodings are tested (conducted in FME) and presented in XML form. Lastly, the findings are presented in a narrative format, structured around principles of the three pillars of a scenario (Context, Strategy and Configuration). The workflow is illustrated in Figure 3.5.

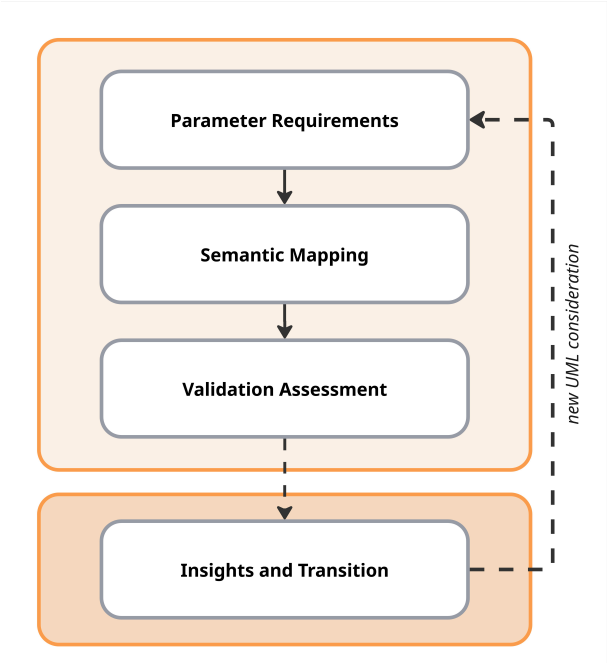


Figure 3.5: Scenario ADE evaluation and refinement process.

The activities are applied as follows:

1. **Parameter Requirements.** Universal schema requirements are derived from the three pillars (Table 1). For each test scenario, concrete data elements that require storing are identified (e.g., "KNMI 2050Hn weather file as URI", "modified wall U-value in W/m²K"). Full tables are presented in the Implementation chapter (Section 4.2.1).

2. **Semantic Mapping ("Re-use First").** The concrete data requirements are mapped to the schema, prioritizing the reuse of CityGML core or the current Scenario ADE. This section highlights the critical mapping gaps found where parameters are forced into unsuitable generic slots or lack a dedicated home entirely.

3. **Validation Assessment.** Stepping back from individual parameters, this assessment evaluates the broader schema architecture. It judges whether the version's conceptual structure reflects the design principles (Table 1) and the theoretical definition of a scenario.

4. **Insights and Transition** The findings from activities 1–4 are concluded and structured around the foundational intentions and design principles per pillar. The insights identify structural capabilities and limitations, referring back to the specific scenario parameters and principles. The section also bridges the lessons learnt from the "refinement" leading to the re-valuation step of the next artifact.

Note: Depth of Evaluation and Micro-Iterations

The depth of the evaluation varies per beta version. First in foremost, it should be mentioned that Beta 4 was an interim project version not seen and not relevant to the Scenario ADE development, therefore it is not considered in this research. For Beta 3 and Beta 5, the evaluation was more focused on theoretical refinement, documenting the ideological shifts that translate into schema architecture changes. This same theoretical evaluation was initially applied to Beta 6. However, as Beta 6 approached a more functional state, aligning more closely with the objectives of the research, the development shifted towards more frequent "micro-iterations". During this time, specific edge cases and validation gaps were identified and refined through more targeted testing. Because these adjustments were often structural fixes rather than broad conceptual changes requiring a full re-evaluation, they are not treated as separate evaluation cycles. Instead, the Beta 6 which is evaluated is the product of the micro-iterations. This iterative process culminates in a single and current version of the schema, which is then subjected to the Phase 4 empirical testing. Any feedback loops, iterations, alternatives and decisions that shaped the Scenario ADE are discussed during the process and insights section provides a summary of each respective beta version.

3.5. Final Encoding and Validation Approach

The final phase translates the refined UML (Beta 6) into its physical encodings and uses concrete encodings both to validate the schema structurally and to test whether the design principles (Table 1) actually hold in practice and that the schema is actually useable. Following the Model-Driven Approach of (Van Den Brink et al., 2013) and (Bachert et al., 2024), the conceptual model is mapped to an XML application schema and to a relational schema for SQL encoding.

For the XML encoding, this involved converting the UML class diagrams drawin in Enterprise Architect (EA)[4] into XSD files through the ShapeChange[5] tool connection. The database implementation and derivation of Data Definition Language (DDL)[6] scripts were created through the Importer/Exporter for 3DCityDB 4.x, where the ADE Manager plugin is used and automatically generates the DDL scripts for the relational schema.

In the case of the DDL script generation, manual refinement is often required to simplify the Object-Oriented (O-O) mapping of the UML to the Entity-Relationship (ER) model of the relational schema. This process involved reducing the amount of output tables, and renaming tables and attributes to be more intuitive. Additionally, many bugs were found and refinements were made throughout this process, ultimately ending at the current version used for Scenario ADE beta 6 testing.

This process although technical, is not the main contribution of this thesis and the method and implementation strategies used are well established and documented in previous ADE developments. More details of the implementation can be found in Agugiaro & Padsala (2025)'s paper.

The actual testing of both respective encodings is an important contribution of this thesis. This is structured in two stages:

3.5.1. Stage 1: Validation of Foundational Operations

Before encoding full neighbourhood-scale scenarios, the schema is first tested on the elementary changes those scenarios are built from. Nine foundational operations are derived directly from

[4] https://sparxsystems.com/products/ea/
[5] https://shapechange.github.io/ShapeChange/4.0.0/
[6] https://www.ibm.com/docs/en/ida/9.2.x?topic=scripts-overview-data-definition-language-ddl

the four test scenarios (S1–S4) and cover root-feature additions, deletions and replacements, the equivalent operations on child features (for example, a solar panel attached to a building), and attribute-level modifications. Each operation is encoded as a small XML instance document and validated against the Beta 6 XSD. The full set is enumerated and demonstrated in the results (Figure 5.8).

Passing all nine validations is treated as a necessary but not sufficient condition: it shows that the building blocks of the test scenarios can be expressed without breaking schema validity, but it does not yet show that the scenarios themselves can be assembled at scale, nor that the schema satisfies its design principles in use. Both are addressed in Stage 2.

3.5.2. Stage 2: Scenario Encoding and Database Integration

The four use-case scenarios (Future Climate, Refurbishment, PV Adoption and Urban Greening) are encoded at neighbourhood scale through Python scripts that extend from the foundational operations, applying changes at scale. A high-level overview can be found in Figure 4.17. A final combined file groups all four scenarios together to test the schema's support for branching lineage from a shared baseline.

The same scenarios are then loaded into a 3DCityDB instance populated with the Beta 6 relational schema. The database stage tests whether the design principles (Table 1) are represented in the schema.

The outcomes of both stages are reported in Section 5. As there are many seperate examples, the results are presented in the form of a directory, pointing to where the worked coding examples can be found. At each example an indicaton of which design principle (Table 1) it puts to the test is added. The reader's guide (Table 14) points to the example that answers a given question, and the same examples are then re-tested inside the 3DCityDB.

4. Schema Development and Evaluation

This chapter introduces the implementation of the iterative design research methodology described in Chapter 3. First, it details the initial simulation runs, comparisons, and parameter analysis. Second, it translates these findings into scenario-specific requirements to refine the final conceptual model of the Scenario ADE. Then the iterative design science research methodology with iterative schema iteration and refinment is applied. Here the requirements and refinement toolkit steps are applied documenting the evaluation process leading to final Beta 6. Lastly, the encoding process of the final Scenario ADE schema to its respective XML and SQL encodings is presented.

4.1. Empirical Analysis of Urban Energy Simulations

This phase implements the empirical analysis defined in Section 3.2. Within the DigiTwin4PEDs project, many comparative simulation work has already been carried out across these neighbourhoods for contrasting future weather files and quantifying the cooling effect of added street trees (Rahmawati, 2025)[7]. Reproducing or extending that work is neither necessary nor the contribution and focus of this thesis.

Instead, the simulations in this phase are explorative and serve as a starting point in the development of the Scenario ADE. The Scenario ADE must store the inputs that define a simulation scenario and the outputs it produces. Therefore, designing that schema requires those inputs and outputs to be observed concretely rather than assumed. Phase 1 runs a small, and deliberate set of SimStadt simulations and treats them first as a means of characterising the structure of simulation data (what a physics-based urban energy simulator consumes, what it adds internally from its own libraries, and what it returns). The principal deliverable of this phase is the structured input/output picture, which Section 4.2 develops into concrete schema requirements.

A secondary outcome is a small set of empirical observations. Several runs probe questions the earlier project work did not examine in isolation, this includes the influence of the solar radiation model, an in-depth comparison of measured against archetype window-to-wall ratios (WWR), and an outlier inspection of per-building demand.These yield interesting insights and, if substantial enough can confirm which inputs and outputs genuinely do influence the results and therefore should be represented in the schema. The actual numerical findings of the radiation model comparison, WWR and outlier analysis are reported in the results chapter (Section 5.1).

4.1.1. Study Area and Datasets

As all of the simulations are grounded in the context of Rotterdam, the following figure provides an overview of all the case-study neighbourhoods involved (Figure 4.1).

[7]Unpublished Master thesis also available @ https://resolver.tudelft.nl/uuid:bf97005e-bd6c-4d76-b5e8-91a94c40d99f



Figure 4.1: Case-study neighbourhoods in Rotterdam.

The 3D city models are CityGML LOD2 datasets produced by preprocessing workflow adapted from Gao et al. (2025), carrying BAG identifiers, year of construction and building usage function code. For Feijenoord and Prinsenland specifically, they are further enriched with per-building per-thematic surface window-to-wall ratios measured by Gemeente Rotterdam, which makes it possible to compare generalised archetype assumptions against actual building characteristics.

4.1.2. Characterising the Simulation Inputs and Outputs

Every simulation in this phase is a variation of the baseline run. A baseline simulation is produced for each neighbourhood using the Perez radiation model, the measured (AI) WWR dataset where possible, and the Meteonorm Rotterdam Monthly TMY3 weather file. It represents the current situation with no interventions, serves as the reference state against which all sensitivity variants are compared, and forms the departure point from which the four test scenarios of Phase 2 branch.

Before any parameter is varied, the first analytical step of Phase 1 makes the baseline run itself explicit. Figure 4.2 gives an input/output view of a single baseline SimStadt run: the data and configuration the user must supply, the archetype data and internal derivations SimStadt contributes from its own libraries, and the per-building results and run metadata it returns.

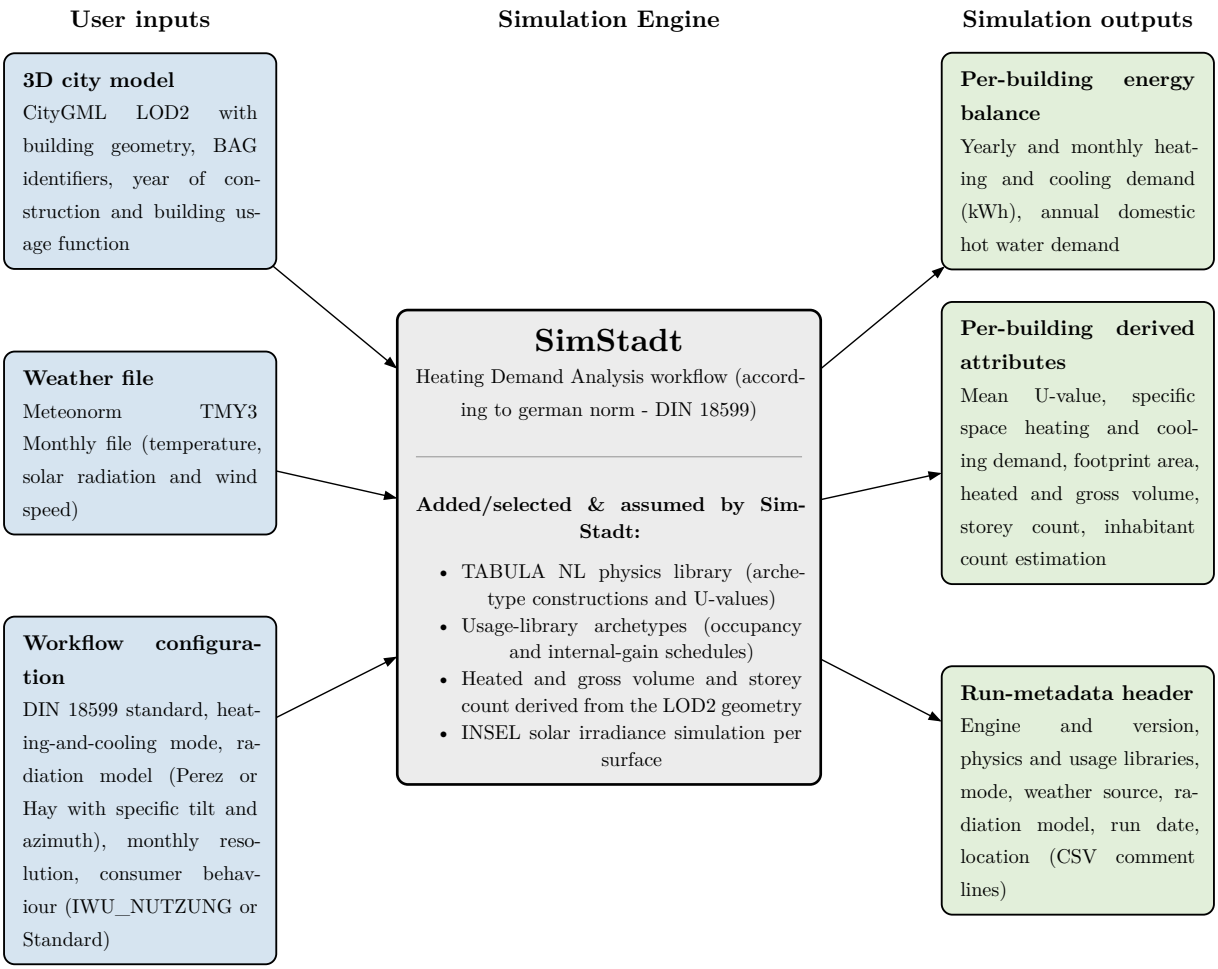


Figure 4.2: Input/output view of a standard baseline Energy Demand SimStadt Workflow run for a given 3D city model.

The baseline run in Figure 4.2 uses SimStadt's Heating Demand Analysis workflow. However, as part of the simulation runs two further workflows are added to the baseline: the PV-potential analysis and heating demand with building refurbishment strategy. Although all three share the same baseline engine, enriched 3D city model, weather inputs and baseline settings each has its own additional configuration parameters and return slightly different output results and metadata. Figure 4.3 therefore overlays the two additional workflows with the baseline kept parameters. The PV-potential workflow moves the result granularity from the building to the individual related surface, while the refurbishment workflow outputs the energy balance (heating and cooling) after altering the thermal envelope as well as the derived the post-refurbishment U-values that define the intervention at surface level as well.

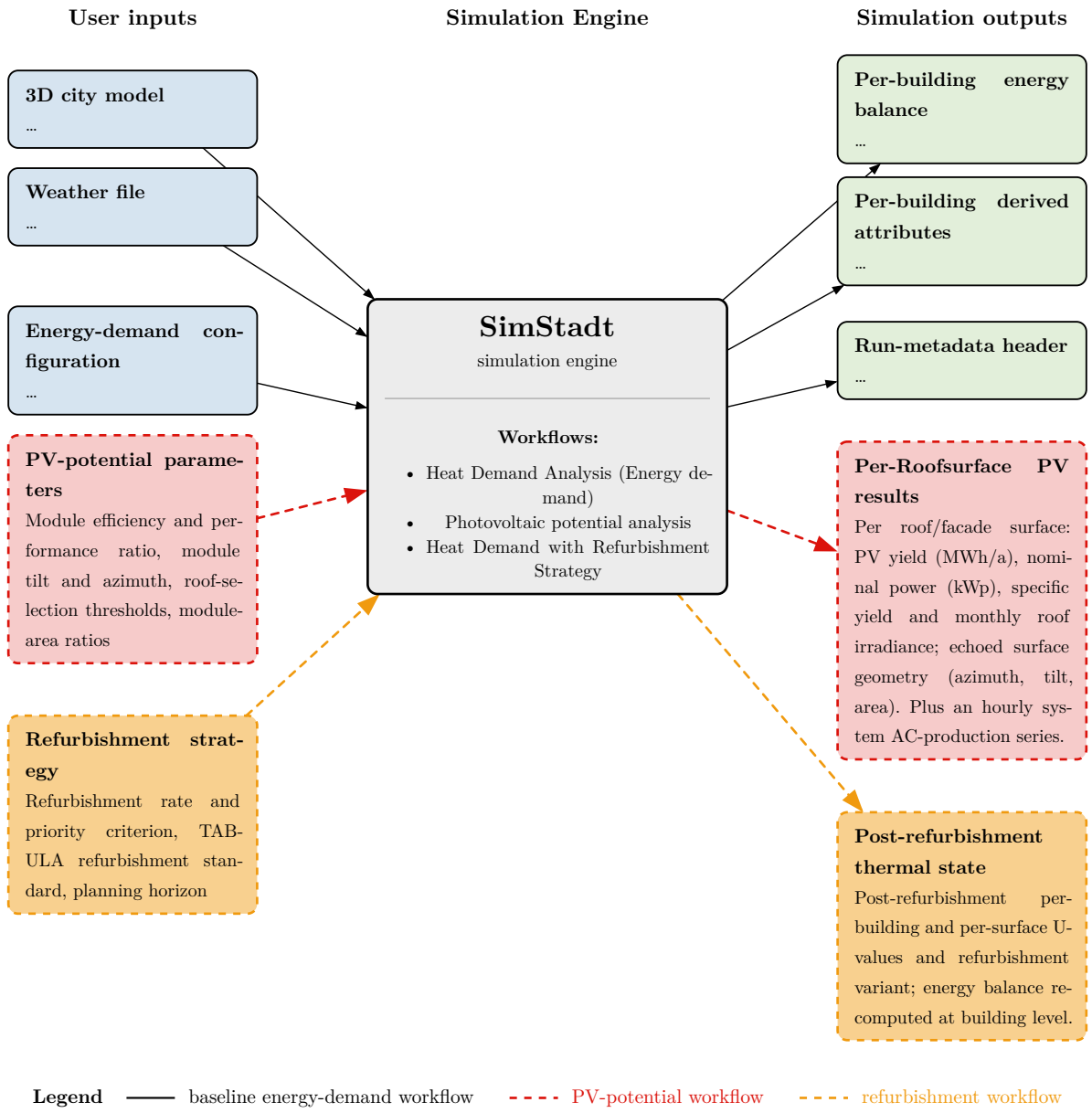


Figure 4.3: Input/output view of the three SimStadt workflows used in this thesis, overlaid on the baseline.

> **Note**
>
> All three workflows write a run-metadata header, with its contents varying depending on the workflow. As it is considered that if the initial metadata can be supported in the Scenario ADE correctly and stored in the right place, then additional metadata can simply be appended using the same logic. Therefore knowing all potential output metadata for simulation engines and run is not relevant for the development of the Scenario ADE as long as it is proven that it functions as intended with a sample.

4.1.3. Exploratory Simulation Runs

With the baseline run and its workflows characterised, the analysis branches into a small family of exploratory runs. Each run uses the same input/output structure of Figure 4.2 or Figure 4.3 if specified, with a single input replaced and all others held at their baseline values. This allowed the further analysis to identify any observed change in the outputs attributable to the parameter

that was varied. The purpose was to see concrete output data, which if inputs visibly move which outputs and therefore if the schema should store them explicitly. The full set of runs is shown in Figure 4.4.

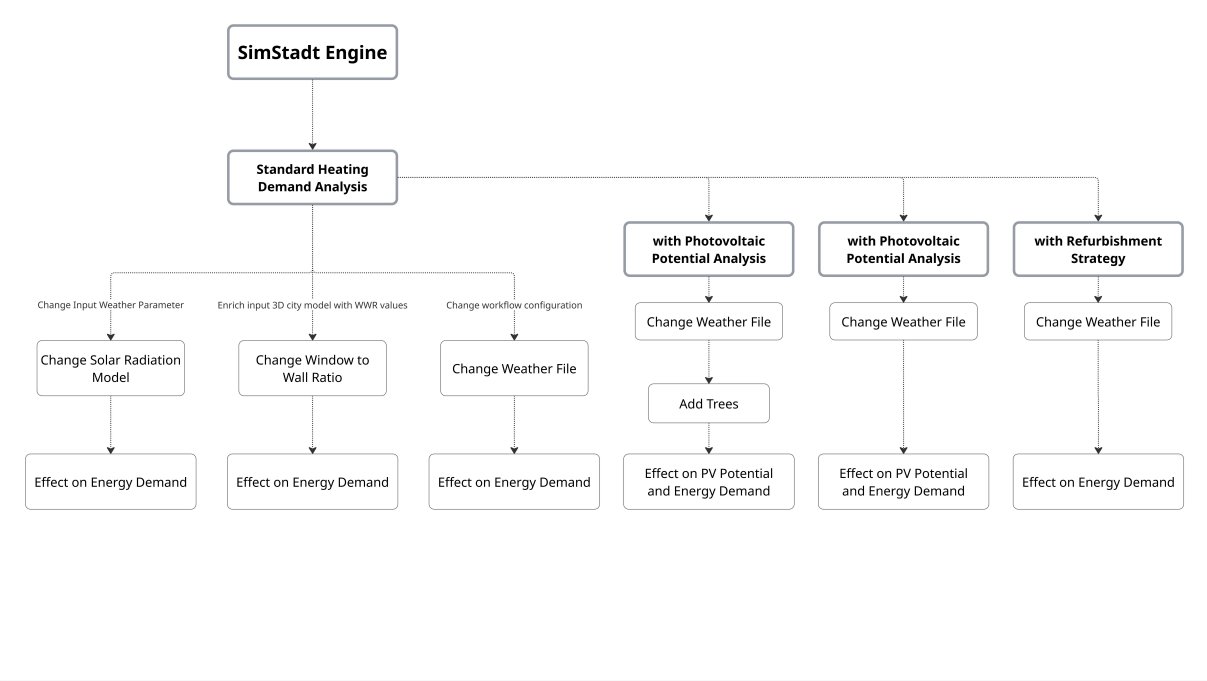


Figure 4.4: The exploratory simulation runs, Standard Heating Demand Analysis runs varying one input from the baseline, and runs with the PV-potential and refurbishment workflows added.

All runs use SimStadt's energy-demand workflow under the DIN 18599 standard, in the heating-and-cooling mode, and at monthly resolution. The building physics library (TABULA Netherlands) and the behaviour profile (IWU_NUTZUNG) are held fixed throughout. The neighborhoods involved covers the intent of each run. The radiation-model comparison is run for all available neighbourhoods, the window-to-wall ratio comparison and the outlier inspection use Feijenoord and Prinsenland as they are the only ones with that data, and the remaining runs use Feijenoord. This is sufficient since the aim is to observe the output structure rather than to compile a result sets. Feijenoord is chosen as it will become the neighborhood on which the results will be encoded. The two parameter families varied across runs are listed in Table 2.

Parameter family	**Baseline value**	**Alternatives tested**
Solar radiation model	INSEL Perez	INSEL Hay
Window-to-wall ratio	AI measured (Gemeente Rotterdam)	7 % (avg of Feijenoord), 10% (avg of Prinsenland), 20 % (Standard WWR archetype)

Table 2: Parameter families varied in the Phase 1 further analysis.

4.1.3.1. Solar Radiation Model

The solar radiation model controls how SimStadt translates weatherfile irradiance into the irradiance reaching each building surface. The baseline uses the Perez model and a comparison run substitutes the Hay model with all other inputs fixed. Because the influence of the radiation model had not previously been isolated within the project, this comparison is run across all available neighbourhoods. It tests whether the radiation model is a configuration parameter that heavily influences the results. The findings are reported in Section 5.1.

4.1.3.2. Window to Wall Ratio Analysis

The window to wall ratio (WWR) is normally inherited from a building-usage archetype rather than measured and is a standard value anywhere between 10% to 50% (Dang et al., 2024). The measured dataset makes a direct comparison possible, and because it represents a genuinely new input it is examined in more depth than the other runs. The baseline uses the measured (AI) values, and compartive runs using standardized percentages approximated the mean WWR of both respective neighborhoods (7 % for Feijenoord, 10 % for Prinsenland) and a commonly used conservative archetype estimate(20 %). The findings are reported in Section 5.1.

4.1.3.2.1. Window to Wall Ratio Outlier Analysis

This analysis identifies which buildings are most affected when a standardised archetype WWR replaces the measured (AI) values. For each building in Feijenoord and Prinsenland, the annual demand of a standardised WWR variant run is compared against the actual measured WWR baseline run, and the relative change is calculated for each demand type $d \in \{\text{heating}, \text{cooling}\}$:

$$\Delta_d = \frac{D_d^{\text{std}} - D_d^{\text{AI}}}{D_d^{\text{AI}}} \times 100\% \tag{4.1}$$

where D_d^{AI} and D_d^{std} are the annual demand of type d under the measured and standardised WWR respectively. Buildings with an extreme Δ_d are flagged using the interquartile-range (IQR) method. Let Q_1 and Q_3 be the first and third quartiles of the values $\{\Delta_d\}$ over all buildings in a neighbourhood, and let the interquartile range be $\text{IQR} = Q_3 - Q_1$. A building is flagged as an outlier for demand type d when its Δ_d lies outside:

$$\Delta_d < Q_1 - 1.5 \cdot \text{IQR} \quad \text{or} \quad \Delta_d > Q_3 + 1.5 \cdot \text{IQR} \tag{4.2}$$

Buildings falling outside these boundsare ranked by $|\Delta_d|$ and the largest cases are inspected visually. This allows us to understand if there are any patterns in the most affected buildings. Relating back to the development of the schema, applying this specific analysis also motivates the need for a Scenario ADE to be able to ensure traceability of results back to the input 3D city model and configurations that produced it. Results are reported in Section 5.1, with the full per-neighbourhood outlier tables given in the appendix (Section C.2).

4.1.4. From Exploratory Runs to Test Scenarios

From all the runs conducted, a second group of runs can be derived where their inputs all include the changed weather file and use an additional workflow. These are combined into the what-if situations as they change weather File to a future-climate context, a refurbishment strategy, rooftop PV potential, and added street trees. As they also provided results, these are compared against each other, and presented at the end of the phase 1 results (Section 5.1.4). These simulations runs, illustrated by the color coding, will be elevated to the four test scenarios that evaluate the Scenario ADE (Figure 4.5).

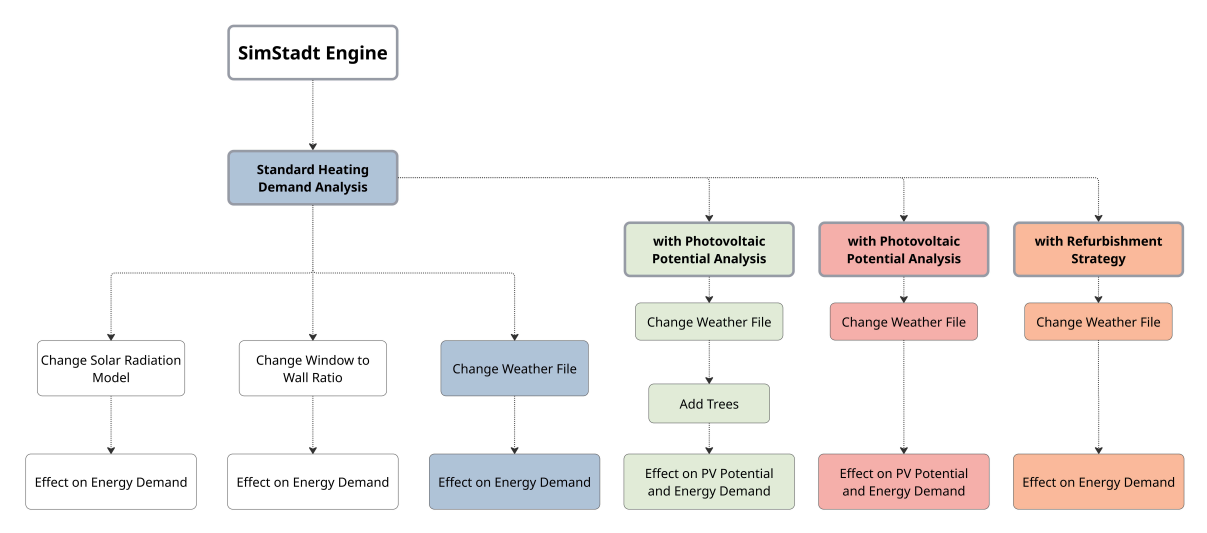


Figure 4.5: The simulation runs carried forward as the four test scenarios used to evaluate the Scenario ADE.

4.1.5. Implications for Schema Requirements

The implications from the runs revealed *which data a Scenario ADE must hold*, and produced empirical findings on *how individual parameters influence the results*. The findings themselves are reported in the results chapter (Section 5.1). The actual implementation from each run is used to motivate schema requirements allowing it to be grounded in observed results rather. A final package diagram presenting an initial idea of how the Scenario ADE relates to what is learnt from the simularion runs in presented in Figure 4.6.

From the results and explicitely deriving what a simulation run requires and produces, the following understandings required in the next phase are made:

1. **Simulation inputs must be stored and linked.** A small configuration choice such as the solar radiation model used produces a noticeable difference in heating and cooling demands (Section 5.1). If these inputs are not recorded, differences in outcomes amongst scenarios from the same source are difficult to explain. The schema must therefore record the parameters of each simulation run and link them to the relevant scenario.
2. **Data must be traceable to specific features.** Substituting measured window-to-wall ratios (WWR) with uniform archetype assumptions substantially and unevenly changes the energy demand across buildings (Section 5.1). The baseline city model might not store this building specific data when it is assumed or derived. The schema must therefore store any new data enrichment explicitly and link it back to the specific buildings it modifies.
3. **Each result must be linked to what produced it.** To investigate outliers or make meaningful comparisons, users must be able to trace a result back to its source (Section 5.1). The schema must therefore link every output to the specific feature, city-model version, and simulation run that produced it.
4. **Results must be stored at their native level of resolution.** The input/output characterisation of the three workflows (Figure 4.2, Figure 4.3) shows that outputs are returned at different levels. These are per building (heating and cooling demand) and per surface (PV yield and roof irradiance). The schema must therefore store results at whichever level they are produced, and keep them distinguishable and attached to their correct features.

These four findings, combined with the scenario review will form the foundation of the three-pillar schema and the concrete data requirements we set for each scenario. While the simulation runs dictate the data requirements (Members, Parameters, Outputs), the scenario literature dictates the need to track their evolution over time (Version Changes). These are represented in the package diagram below (Figure 4.6). Together, they directly inform the design principles carried into Phase 2 (Table 1).

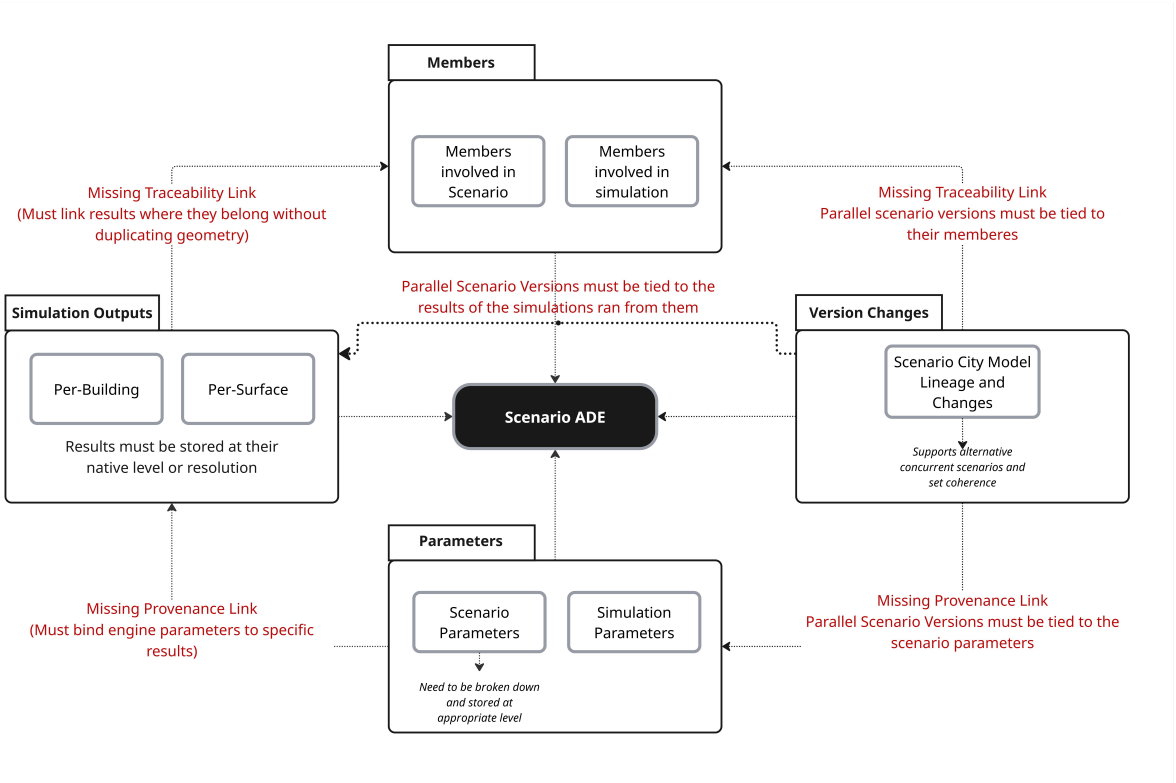


Figure 4.6: Package diagram summarising the findings of Phase 1 and the initial understanding of what a Scenario ADE must be able to represent.

4.2. Scenario Requirements Development

Based on the phase 1 findings, we can inform our parameter choices for the 4 test scenarios presented in Table 3. Each scenario is obtained by dissecting the Phase 1 simulation runs into the three scenario pillars, assigning every observed parameter to the Context, Strategy, or Configuration it belongs to.

ID	Scenario Name	Context	Strategy	Configuration
S1	Future Climate	KNMI 2050Hn	None	Baseline SimStadt Energy Demand Configuration
S2	Refurbishment (Future)	KNMI 2050Hn	Refurbishment	Added refurbishment step and parameter inputs
S3	PV Adoption	KNMI 2050Hn	PV	Baseline SimStadt PV Analysis Configuration
S4	Urban Greening	KNMI 2050Hn	Planting Trees	Added trees to citymodel, PV analysis step and Energy Configuration

Table 3: Overview of the four test scenarios used to evaluate the Scenario ADE.

4.2.1. Scenario-Specific Parameter Requirements

For each of the case study scenarios, select concrete requirements are derived from the established input and outputs of the simulation runs of which they are composed. These requirements are linked to the design principle requirement they refer to presented in the methodology Table 1.

The chosen data parameters are chosen to be a subset of specific parameters, inputs and outputs observed in the simulation runs (and structure presented in Figure 4.3). It should provide a representative picture of the kind of data the Scenario ADE should be able to track and represent. If it functions on these subsets, it should also function on further enriched scenarios.

4.2.1.1. S1: Future Climate

The first scenario is the future climate scenario that represents the current situation with the future climate conditions (KNMI Klimaatscenario 2050Hn). This scenario branches from the baseline city model, replacing the input climate context with a interpolated KNMI2050Hn weather file. The context is the sole varying scenario parameter.

Through this scenario we attempt to test wether the Scenario ADE can handle two different versions, differing by one contextual parameter.

ID	Concrete Data Parameter	Pillar	Design Principle
S1-01	EPW weather description (KNMI 2050Hn future climate) and weather file path/URI	Context	C1, C3
S1-02	Planning horizon year: 2050 (Date)	Context	C2
S1-03	Human readable scenario description (String)	Context	C2
S1-04	Simulation engine name and version (String)	Config	CF1
S1-05	Simulation calculation parameters (various data types, standard identifier, Irradiance Model, etc.)	Config	CF1
S1-06	Heating and Cooling demand per building (per year and monthly) (Double and List, kWh/a)	Config	CF2, CF3
S1-07	Null strategy declaration (zero intervention)	Strategy	C1

Table 4: Scenario 1 Mapping: Concrete data parameters to design principles.

4.2.1.2. S2: Refurbishment under Future Climate

The refurbishment under future climate scenario branches from the future climate scenario. It applies a refurbishment strategy to the baseline model under a horizon of now till 2050. It serves to primarily test feature and attribute modifications, it allows the schema to see whether it can store the new derived renovation values at their level of resolution as well as the new results created from those derivations.

ID	Concrete Data Parameter	Pillar	Design Principle
S2-01	Same climate and horizon year as Scenario 1	Context	C1
S2-02	Modify Wall, Window and Roof thermal U-value (Double, W/m²K)	Strategy	ST1, ST2
S2-03	Refurbishment standard identifier ("TABULA Advanced")	Strategy	C3
S2-04	Specific ability to only modify values	Strategy	S1
S2-05	Simulation engine name and version (String)	Config	CF1
S2-06	Simulation calculation parameters (same as S1 + Priority Criteria and relevant parameters)	Config	CF1
S2-07	Heating and cooling demand post-refurbishment (yearly and monthly)	Config	CF2, CF3

Table 5: Scenario 2 Mapping: Refurbishment parameters to design principles.

4.2.1.3. S3: PV Adoption under Future Climate

The PV Adoption under future climate scenario is also branching from scenario 1 and applies a PV adoption strategy to the baseline model under the same future climate weather. It serves to test whether the Scenario ADE can handle the additional of new features which may be children of existing features. It also serves to further test storing generation results and not demand results. The PV potential results will need to be stored a roof level.

ID	Concrete Data Parameter	Pillar	Design Principle
S3-01	Same climate and horizon year as Scenario 1	Context	C1
S3-02	Add a new PV system object (geometry and attributes)	Strategy	ST1
S3-03	Simulation engine name and version (String)	Config	CF1
S3-04	Simulation Calculation Parameters (surface area, performance ratio, etc.)	Config	CF1
S3-05	PV potential yield per building roof (MWh/a)	Config	CF2, CF3
S3-06	Monthly irradiance on roof plane (W/m²)	Config	CF2

Table 6: Scenario 3 Mapping: PV addition parameters to design principles.

4.2.1.4. S4: Urban Greening (Planting Tree's for Shade and Cooling Demand Reduction) under Future Climate

The urban greening scenario also branches from scenario 1 and adds new vegetation features (trees) to the baseline model under the same future climate weather. It serves to test whether the Scenario ADE can handle the addition of new standalone features. It also serves to test storing more than one type of output under a single scenario, holding both the PV analysis results and the demand results together.

ID	Concrete Data Parameter	Pillar	Design Principle
S4-01	Same climate and horizon year as Scenario 1	Context	C1
S4-02	Non-Building Feature Addition (Vegetation)	Strategy	ST1
S4-03	Vegetation Specific Attributes (eg: type, height, crown diameter)	Strategy	C3
S4-04	Simulation engine name and version (String)	Config	CF1
S4-05	Simulation Calculation Parameters (Both for PV Analysis and Energy Demand)	Config	CF1
S4-06	Same PV requirements as S3	Config	CF1, CF2
S4-07	Heating and Cooling demand post-greening (yearly and monthly)	Config	CF2, CF3

Table 7: Scenario 4 Mapping: Vegetation parameters to design principles.

4.3. Scenario ADE Evaluation and Refinement Process

This section evaluates three versions of the Scenario ADE (beta 3, beta 5 and beta 6) as an iterative story, using the refinement workflow from Section 3.4.1 to guide the process. Each version is presented in the same four subsections (Overview, Semantic Mapping, Validation Assessment, and Insights and Transition). This also then makes the steps stay comparable and adds structure to what is a iterative and nonlinear process of testing and discovering. During the refinement, usability of the schema is important, testing whether the scenarios can in fact be encoded in the schema in a way which stays true to the principles.

Domain agnosticism is the intention that emerged from the work rather than preceding it (Note: Domain Agnosticism). The lowest level alternative would have been developing an extension

from a base like the CityGML TestADE[8] that does little more than register in the 3DCityDB, and the easiest richer path would have been to build straight on from the Energy ADE, but both reduce the scope of the ScenarioADE once its wider use became apparent during the development. Domain agnosticism is therefore enforced from here on at the beginnig of the evaluation so the schema does not import or depend on any domain ADE and avoids fixed slots for energy specific data. The validation keeps the same mindset, focusing on whether each parameter sits in the right place and stays traceable rather than on small open fields an application layer can restrict on its own.

Tags (F1-F3, C1-C3, S1-S2, CF1-CF3 from Table 1) are cited inline so each observation traces back to the principle it touches. Parameter requirements come from Section 4.2.

4.3.1. Scenario ADE beta 3

The starting point for the refinement process begins with the Scenario ADE beta 3. It was built inside the DigiTwins4PEDs project as an update to the original AIT Scenario ADE (Section 2.5). It was developed without explicit reference to the IL-tradition synthesis or the XSP context and strategy distinction later adopted as the conceptual reference for this thesis (Section 2.1.1, Section 2.1.2).

4.3.1.1. Overview beta 3

Structurally, Beta 3 is small and simple model. It has one core module with four classes: `Scenario`, `ScenarioMember`, `AbstractScenarioModification` and `AbstractGenericParameter`. The UML is shown in Figure 4.7, and in Figure 4.8.

The `Scenario` class is the starting point. It extends `CityObject`, so it has its own ID and can be referenced. On it you can attach a bit of metadata on top of what it inherits from cityObject (a type or code, a validity period, the name of the project the scenario belongs to), a list of the `ScenarioMember` entries that make up the scenario, any modifications that apply to the scenario as a whole, and a `dependsOn` link pointing to another `Scenario`.

The `ScenarioMember` class is what connects a scenario and the city objects it relates to. It extends `gml:_Feature` rather than `CityObject`, so it is not a top-level object on its own and only exists nested inside a `Scenario`. Each instance does three things. It points to exactly one existing city object (a specific building, for example), it states what operation is being performed on it (`addedMember`, `removedMember`, `modifiedMember` or `unmodifiedMember`), and it can carry member-level modifications that describe how that object participates in the scenario.

The modification classes describe what is actually being changed. They share an abstract parent `AbstractModification` which gives them a common type code. Beta 3 has four concrete kinds. `LayeredConstructionModification` points at a domain feature describing a layered construction, `ScheduleModification` points at a schedule, `DescriptiveModification` just records that something was changed, and `PropertyModification` carries a string `attributeRef` plus a typed new value. The first two of these depend on Energy ADE features.

`AbstractGenericParameter` is a DataType inspired by the CityGML Generic module. It exists to attach extra metadata to a `ScenarioMember`. Each parameter has a name, an optional description, and a value of one of several primitive types (string, integer, boolean, double, list of doubles, datetime, measure or URI). It also appears once inside `PropertyModification` as the carrier for the new attribute value. In practice anything the schema does not have a dedicated home for would end up using this mechanism. This is important as the mapping below is largely

[8] https://github.com/3dcitydb/extension-test-ade

involves how most scenario requirement parameters needing a more formal space end up using this mechanism and are strucutally the same.

Lastly, a brief explanation of the relationship between the classes. A `Scenario` composes its `ScenarioMember` entries, so they are part of the scenario. A `Scenario` also aggregates its scenario-level modifications, which belong to it without being strictly bound to its lifecycle. A `ScenarioMember` aggregates its own member-level modifications the same way, and it associates to the city object it targets through a property type that lives independently in the dataset. The `dependsOn` link between scenarios is a flat association. It lets one scenario point at another but says nothing about how they are related.

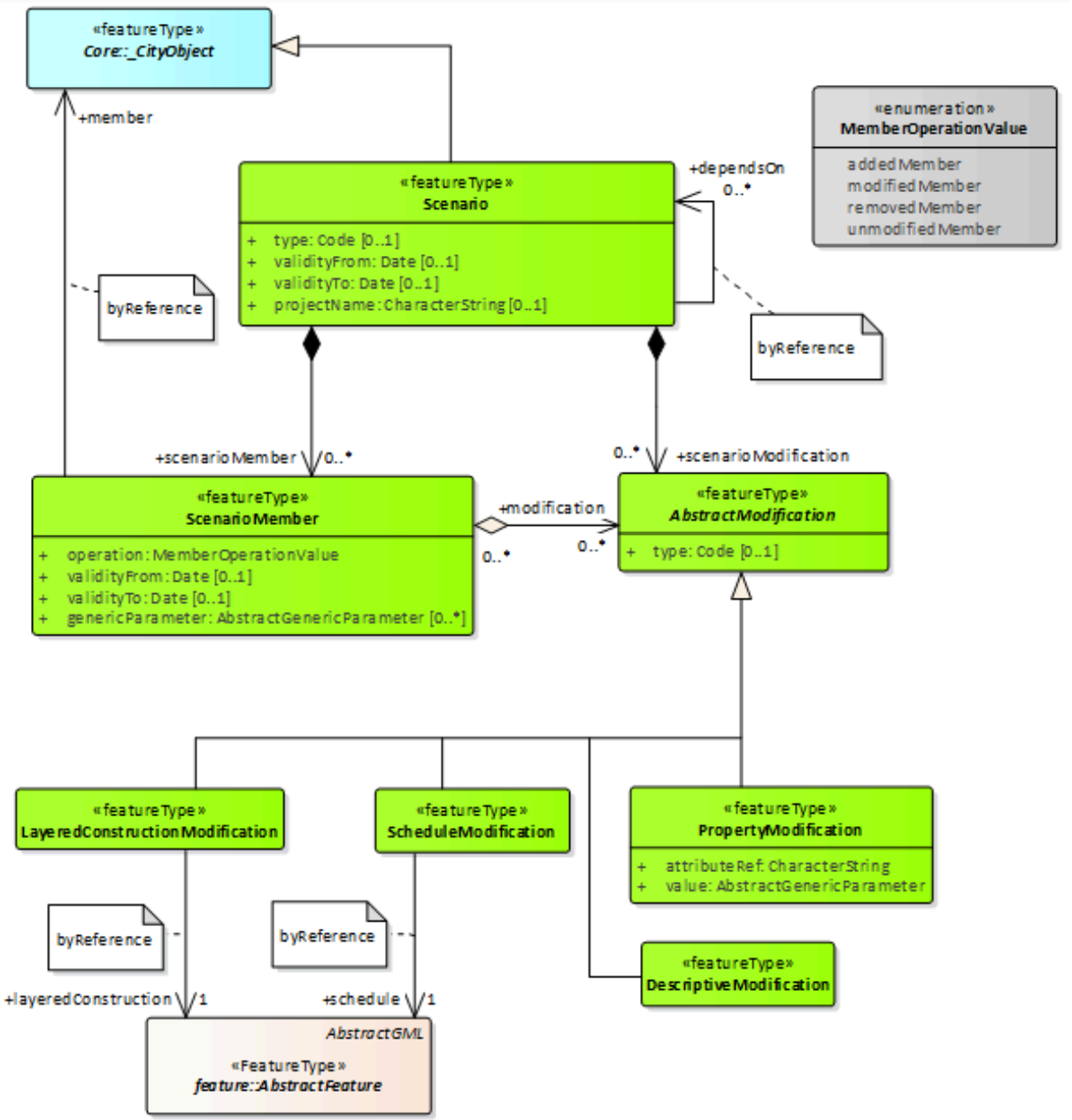


Figure 4.7: Scenario ADE beta 3 UML diagram

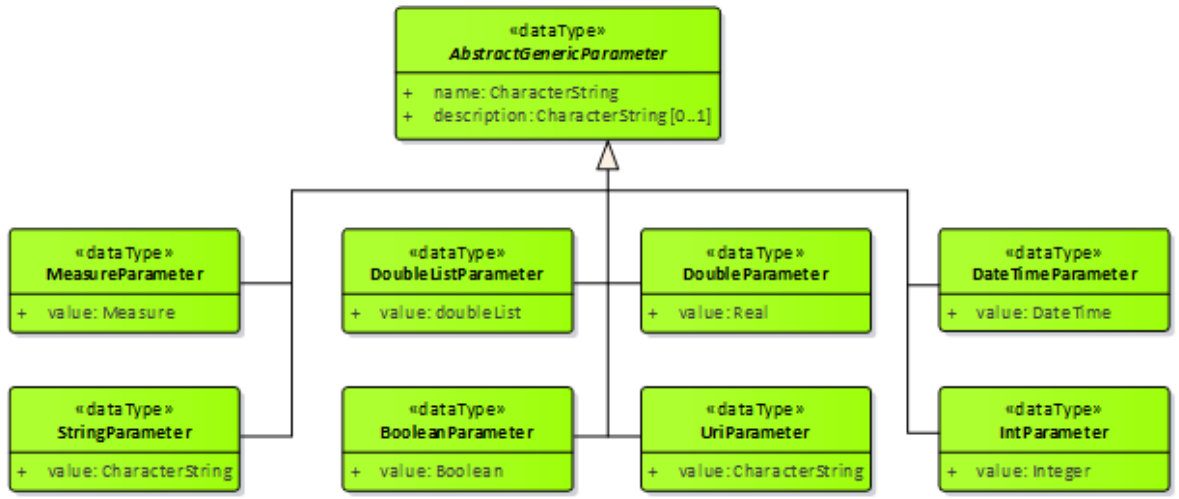


Figure 4.8: Scenario ADE beta 3 *AbstractGenericParameter*

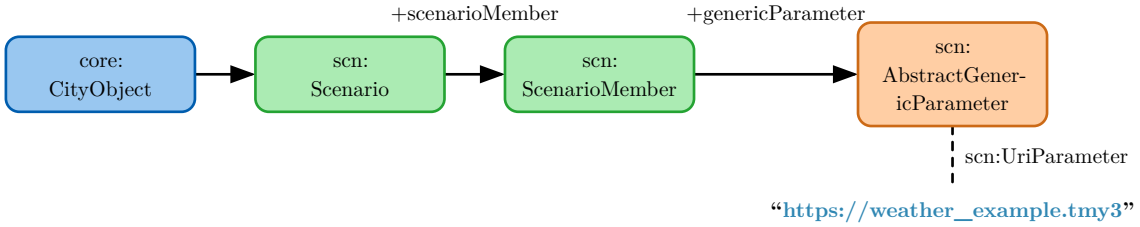
4.3.1.2. Semantic Mapping

Now that the starting ScenarioADE has been introduced and described, the refinement begins, starting with mapping the requirements to the where they would be stored in the schema.

Key findings regarding the parameters are highlighted and discussed in this section. However, when evaluating the beta, a much bigger semantic mapping was conducted for all of the given concrete data requirements. The mapping points brought up below are the important findings which actually led considerations, discussions and decisions involving the next iteration. They are related back the three pillars and the requirement were derived from. If for example the Energy ADE can be used to store the attribute, that option is shown alongside. When possible, the storage path is visualised diagrammatically to improve understanding:

1. **Context parameters have no dedicated space in the schema** *[addresses parameters S1-01(of all scenarios), S1-02; touches C1, C2]*
   - The first mapping concerns where context parameters go. The concrete cases are the Narrative (S1-03), weather URI (S1-01) and the planning horizon (S1-02). CityGML 2.0 core has no climate or weather concepts, so the Scenario ADE falls back on its own `AbstractGenericParameter`. In Beta 3 this slot sits only on `ScenarioMember`, so the weather URI attaches to a member rather than the `Scenario`. The `Scenario` could instead carry a `gen:uriAttribute` directly through its `core:_CityObject` base. The horizon year could map to the scenario validity period, but that period describes when the scenario record is valid, not the future it represents, so it also ends up as a generic parameter. Either way the value stays an undistinguished generic parameter with no slot of its own, a point picked up in the validation. The mapping is shown below.

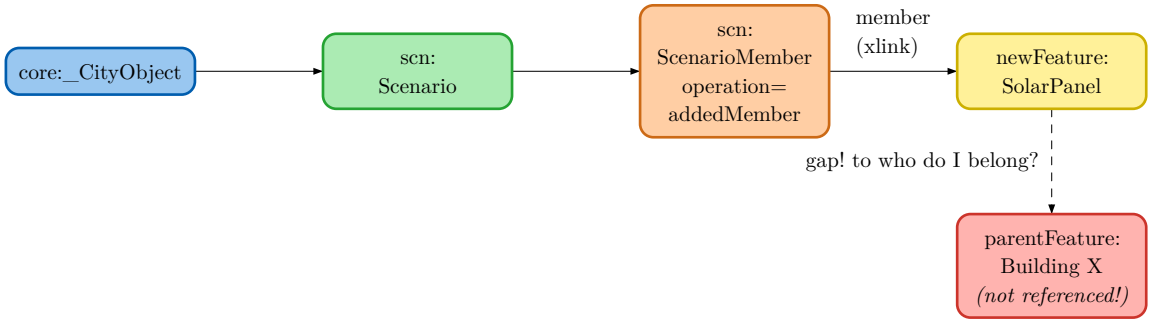
   Beta 3 Solution: `_CityObject` → `Scenario` → `ScenarioMember` → `genericParameter` (with name, description and value) → `UriParameter`



A domain ADE such as the Energy ADE could store a weather file in a structured way. It has a dedicated `WeatherData` class that references a weather file through a time-series file location. As it hooks to the `CityObject`, this file and URI could directly sit on the Scenario. Re-using it would tie the Scenario ADE to one domain though, against the domain agnosticism principle, so it is not adopted. As context is a pillar of the Scenario ADE, another domain scenario should not have to use the EnergyADE in order to store the context data. This should be structurally built into the ScenarioADE.

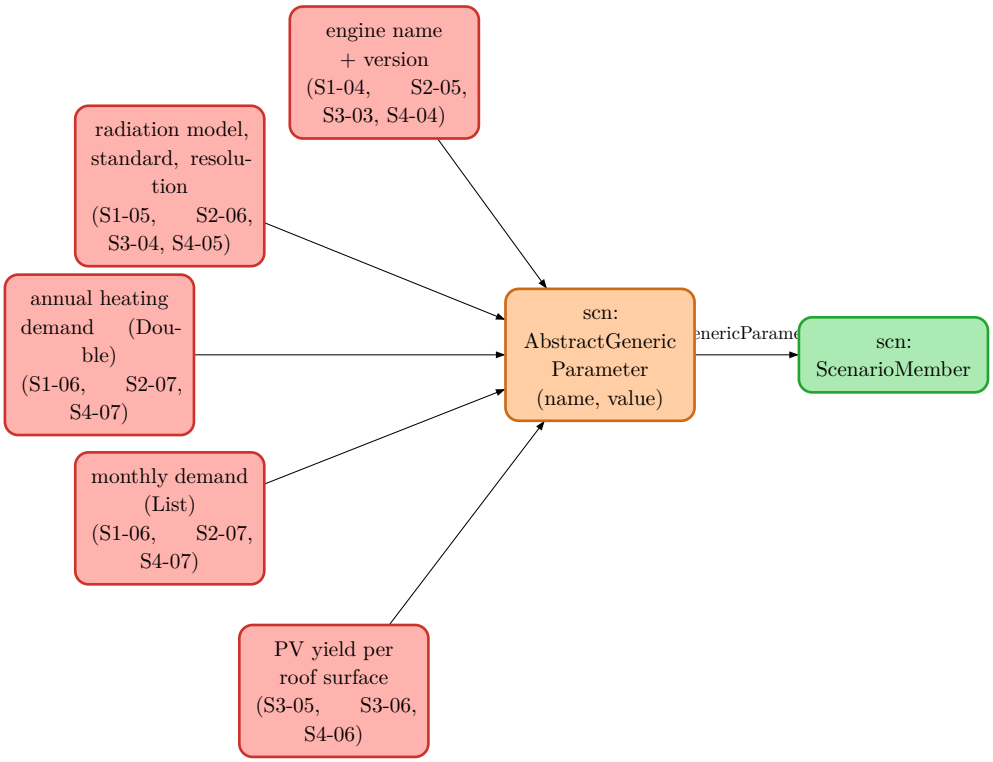
- **Child feature additions cannot be stored in the schema** *[addresses parameters S3-02, S4-02; touches ST1]*
  - The PV scenario adds a new PV object as a child of a roof (S3-02) and the greening scenario adds vegetation as a non-building feature (S4-02). In Beta 3 a `ScenarioMember` with `operation=addedMember` references the new feature through `member`, but there is no slot for the existing feature it attaches to. The added feature is therefore structurally orphaned. Beta 3 can say a feature is added, but not that it is added to Building X, Roof Y. In an XML encoding the new feature ends up ends as an orphaned `CityObject` at root level as no information is provided about its parent.Ultimately, this leads to invalid CityGML files.

Beta 3 Solution: `_CityObject` → `Scenario` → `ScenarioMember` (`operation=addedMember`) → `member` → new feature (parent not referenced)



- **Simulation configuration and results have no dedicated place in the schema** *[addresses parameters S1-04-S1-06, S2-05-S2-07, S3-03-S3-06, S4-04-S4-07; touches CF1, CF2, CF3]*
  - All four scenarios share the same gap. They each list a simulation engine name and version (S1-04, S2-05, S3-03, S4-04), a set of calculation parameters (S1-05, S2-06, S3-04, S4-05), and per-building or per-surface results (S1-06, S2-07, S3-05, S3-06, S4-06-S4-08). Beta 3 has no class dedicated to a simulation engine, no class dedicated to its settings, and no class dedicated to results. A per-building result can technically be attached to a `ScenarioMember.genericParameter`, which through the member's `member` xlink does link it to a specific city object. But the same `genericParameter` slot already carries interventions and context parameters on that member. The schema

cannot distinguish an input from an output, and there is no way to record which simulation engine produced which value. If two engines were run on the same scenario, the resulting `DoubleParameter` instances would be structurally identical.



The diagram above shows how the engine identifier, the inputs and the results all collapse into one `AbstractGenericParameter` slot, with no class distinction between them.

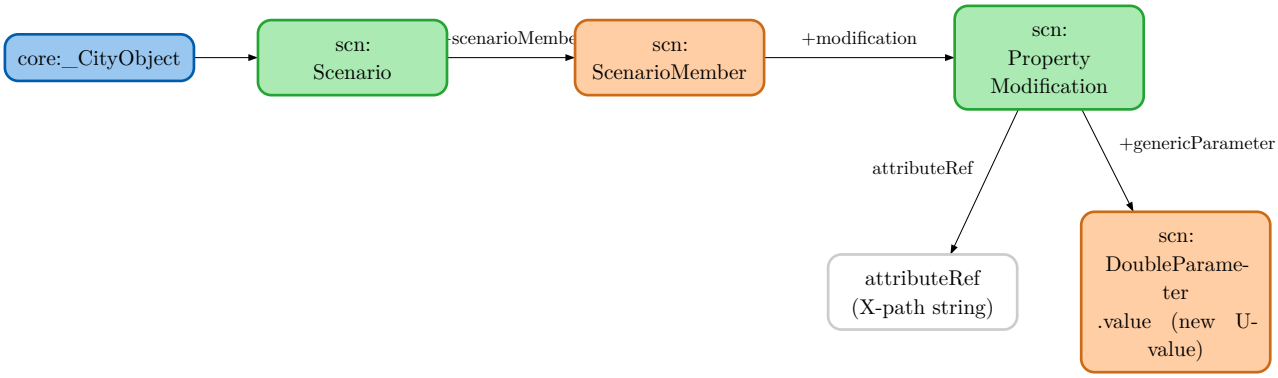
> Beta 3 Solution: No clean solution exists within the Beta 3 schema without the conflation of scenario, simulation, parameters and results.

- In this instance the mapping could also be done through the EnergyADE 3.0, and it offers a `AbstractResource`, attached to any `CityObject` through its "ade-hook", with `AbstractTimeSeries` for monthly or hourly series. However, this approach does have problems, even setting domain agnosticism aside. The resource attaches to the building rather than to the scenario container. If a building participates in multiple scenarios with different heating results, all of those resources pile up on the same building and cannot be traced back to a specific scenario or run. The `resource` hook could instead be attached to the `Scenario` itself, since in Beta 3 the `Scenario` is also a `core:_CityObject`. That moves the result on the scenario class but loses the other half of the link, because an `AbstractResource` carries no reference to the building or surface it describes.

- **Attribute interventions are currently not fully supported** *[addresses parameter S2-02; touches ST1, ST2]*
  - The refurbishment scenario asks to modify the U-value of specific Wall and Roof surfaces (S2-02). The only option is `PropertyModification`, attached to a `ScenarioMember` through its `modification` slot. It carries an `attributeRef` naming what changes and a `genericParameter` holding the new typed value (a `DoubleParameter` for a U-value). This was added to the artifact retrospectively, after the X-path and X-link discussion

stemming from a Beta 5 discovery was had (Section C.1) showed that an just a X-link cannot work. It is useful because it avoids duplicating the geometry. The `attributeRef` is a `CharacterString` with no constraints though, so whether it is an X-path, a property name or anything else is left to convention. This is briefly mentioned in the validation.

Beta 3 Solution: `_CityObject` → `Scenario` → `ScenarioMember` → `modification` → `PropertyModification.attributeRef` (X-path) + `PropertyModification.genericParameter` → `DoubleParameter.value` (new U-value)



4.3.1.3. Validation Assessment

The mapping showed where important parameters should sit on how they currently fit into the schema. The validation assessment now steps back to a more conceptual question. Does Beta 3 let a user work with scenarios the way the three pillars and planning practice intend, and does its structure match what a scenario actually is. The points are ordered from the conceptual gaps that shape the next iteration down to the smaller items.

- **The three pillars cannot be retrieved independently** *(R3.2) [addresses C1, CF1, CF2]*
  - The clearest test of the schema is whether a user can take one scenario and understand its Context, Strategy and Configuration apart from one another. In Beta 3 this is not possible. A narrative, weather file or temporal horizon (context), a wall U-value change (strategy) and a heating demand result (configuration) are all the same semantically ambiguous `genericParameter` on a `ScenarioMember`, carrying only a name and a description. Because `Scenario` inherits from `core:_CityObject`, it also gains CityGML's native generic attributes (`gen:stringAttribute`, `gen:uriAttribute`, `gen:doubleAttribute`) through the `_GenericApplicationPropertyOfCityObject` hook, so context can sit on the scenario itself, just as an undistinguished generic attribute. Beta 3 therefore carries two parallel generic mechanisms, but not the schema architecture which should structurally reflect the scenario strategy decomposition proposed.
- **Scenarios cannot express how they relate** *(R3.4) [addresses Scenario Set Coherence (F2)]*
  - Based on the literature and use case, urban planners build scenarios to test different strategies which are equivalent to scenarios branching off from a baseline and planned in sets. A lot of value and importance comes from the ability to compare them and observe their relative differences. Beta 3 expresses this only through `dependsOn`, which states that one scenario depends on another without saying how (inheritance, alternative, baseline) or what changed. Lineage is left to convention and falls back on

free string parameters, which a user or an application layer cannot use to reconstruct the structure of all scenarios in a file or database. The same use cases expose two concrete needs that recur through the refinement. An added feature (a PV panel, a tree) needs to record its parent feature, and an attribute change (a wall U-value) needs to happen without duplicating the whole feature.

- **Other small findings along the way** *(R3.1, R3.3) [addresses ST1, ST2]*
  - `PropertyModification.attributeRef` is an unconstrained `xsd:string`, which the intended use is to use with a Xpath. At the moment, the schema cannot check the spelling, format or if what is referenced actually exists or is the intended target. Out of a principle to allow as much customiseability as possible, this is noted, however thenaming check should be left to the application layer. Separately, the `MemberOperation` enumeration cannot express a single replacement, so replacing a wall or a PV panel needs a redundant `addedMember` plus `removedMember`. This is a small enumeration addition rather than a conceptual gap.
- **A scenario inherits some unused attributes** *(R3.5) [addresses C2, lineage across S1-S4]*
  - As a smaller, final point, because `Scenario` inherits from `core:_CityObject` it also picks up attributes with no meaning for a scenario (`relativeToTerrain`, `relativeToWater`, `externalReference`) and duplicates the lifecycle attributes with its own "real-world" time (`creationDate`, `terminationDate`) next to the scenario's own validity period. This is a design decision more than a real shortcoming. If a scenario does not need these attributes it simply does not use them.

Below is a table which briefly summarises the validation assessment discussion points and provides a potential recommendation, bringing the schema closer to a functional Scenario ADE (Table 8)

Code	Addresses	Recommendation
R3.1	S2	Precise attribute name checking is in the scope of the application layer. An XSD pattern facet[9] could restrict the value strings which follow the shape of X-Paths but there is not way to know if the attribute exists. An enumeration of allowed names could also be done, but is too constraining and would conflict with the domain agnosticism principle and reduce potential useability.
R3.2	C1, CF1, CF2	Introduce dedicated classes where contextual parameters, simulation results, and interventions can be kept, rather than collapsing all three into the same `genericParameter` slot on `ScenarioMember`.
R3.3	ST1	Expand `MemberOperationValueType` with a `replacedMember` value, so that a single member can express replacement without using `addedMember` plus `removedMember` redundantly.
R3.4	Scenario Set Coherence (F2)	Formalise the `dependsOn` association to express how scenarios relate to each other to make them more useful for planners. This would also greatly benefit a application layer retrieve related scenarios and compare results.
R3.5	C2, Lineage (F2)	The scenario classes can move from `core:_CityObject` to `gml:_Feature`, which removes the attributes that do not match what a scenario is. The more useful change is to inject lifecycle attributes (`versionId`, valid period, scenario back-reference) into every `_CityObject` through an "ade-hook" on `_GenericApplicationPropertyOfCityObject`, so versioning becomes universal across all CityGML thematic classes without subclassing each one. This also aligns the 2.0 ADE with the `AbstractFeatureWithLifespan` pattern introduced natively in CityGML 3.0, easing a future schema upgrade.

Table 8: Summary of Beta 3 validation recommendations.

4.3.1.4. Insights and Transition

The per pillar summary of the insights for Beta 3 is as follows:

Context

- Beta 3 has an absence of structural boundary between the external context, and the strategy (interventions) applied within it. For example, a KNMI 2050Hn weather file name or URI (S1-01) and a wall U-value modification (S2-02) are identically encoded as `genericParameter` elements living on a `ScenarioMember` or `Scenario` through the CityGML Generics Module. Energy ADE solution for the URI file exists on the Scenario, however this is against domain

[9]Example 4.1.2.1: https://www.w3.org/TR/xmlschema-2/#derivation-by-restriction

agnosticism (F1). This distinction should be structural, at the moment it leads to a issue where a machine, or even human reader will struggle to understand what is what.

Strategy

- Beta 3 tackles "which object" and "what happened" on the single `ScenarioMember` level.
- `PropertyModification` prevents duplicated geometries, however unconstrained attribute references can causes ambiguity and should be restricted. This should be considered in the later development of the Scenario ADE.
- Specific modification types (`LayeredConstructionModification` , `ScheduleModification`) conflict directly with domain-agnosticism by depending on Energy ADE. These should be removed in favour of keeping the scenario ADE domain agnostic.

Configuration

- Simulation configurations and results mapping leave a gap which currently is unresolved (CF1, CF2, CF3 unmet). No class or mechanism links outputs to the specific city object producing them.
- Storing results as `genericParameter` on `ScenarioMember` conflates the scenario definition with specific simulation run outputs, making distinguishing what produced what results impossible.
- Configuration and results should be stored on the scenario level, with a clear link of what produced them and where they belong in the citymodel. For example, while using EnergyADE's `AbstractResource` , requirements can be stored but on object instead of the scenario, losing context if a building participates in multiple scenarios.

Transition to Beta 5:

Findings from the Beta 3 iteration demonstrate a shift toward a more generalized approach designed to structurally match the intended use case. Structural modifications include the introduction of dedicated `ScenarioLibrary` and `SimulationLibrary` containers, (a structural step towards seperating the simulations from scenarios C1 and CF1) and the removal of all Energy ADE modification classes to establish domain agnosticism (F1). Other small findings support minor adjustments, such as adding `replaced` to the operation enumeration, due to it's relevance in the evaluated use case scenarios.

As the iteration and refinement process remains in an experimental phase, resolving the `depends on` issue from beta 3 is not yet the primary focus. This issue is documented, but the iterative testing process dictates that changes must be made gradually rather than all at once. Current findings will be re-evaluated and integrated later, provided the broader schema architecture proves to be functional. For example, the last version (beta 6) uses an xlink plus `parentFeature` link rather than the `attributeRef` and X-path method. This choice provides a cleaner integration with the current model, as the high level of precision offered by X-path was at that stage no longer deemed necessary.

4.3.2. Scenario ADE beta 5

As mentioned in the methodology, the evolution skips Beta 4. This was interim project version not seen and which did not contribute to the further development of the schema.

From the evaluation of beta 3 and further intentions towards narrowing the gap between the scenario pillars and the schema, beta 5 arises. This transition can be seen as the first test where a step towards a more generalised approach is taken, and where domain dependency is removed and the need for simulation configuration storage is addressed.

4.3.2.1. Overview beta 5

Beta 5 turns the Beta 3 model into a larger schema split across two modules. The core module supplies the lifecycle and library, and the scenario module holds the scenario and simulation logic. New classes in this iteration are `AbstractADEFeature`, `AbstractFeatureWithLifeSpan`, `AbstractLibrary`, `ScenarioLibrary`, `SimulationLibrary`, `Simulation`, `AbstractMember` and `SimulationMember`. The modification classes from Beta 3 (`AbstractScenarioModification` and its four subclasses) are dropped entirely. The UML is shown in Figure 4.9 for the core module and in Figure 4.10 for the scenario module.

Inspired by CityGML 3.0, and following the recommendation R3.5, the lifecycle pattern is introduced through `AbstractADEFeature` and `AbstractFeatureWithLifeSpan`. `AbstractADEFeature` carries a `creationDate` and a `terminationDate`. `AbstractFeatureWithLifeSpan` extends it with an `identifier`, a `validFrom` and a `validTo`. Every new ADE feature inherits from this pattern.

`AbstractLibrary` is a new featureType that extends `core:_CityObject`. It carries a `type` code, a `source` and an `author`. It is the parent of two concrete `{root}` containers, `ScenarioLibrary` and `SimulationLibrary`. Both are intended to sit as top-level members of a CityModel. The first holds `Scenario` instances and the second holds `Simulation` instances. Their reason for being is to keep scenario definitions and simulation runs separate from the physical city objects of the underlying model, and from each other.

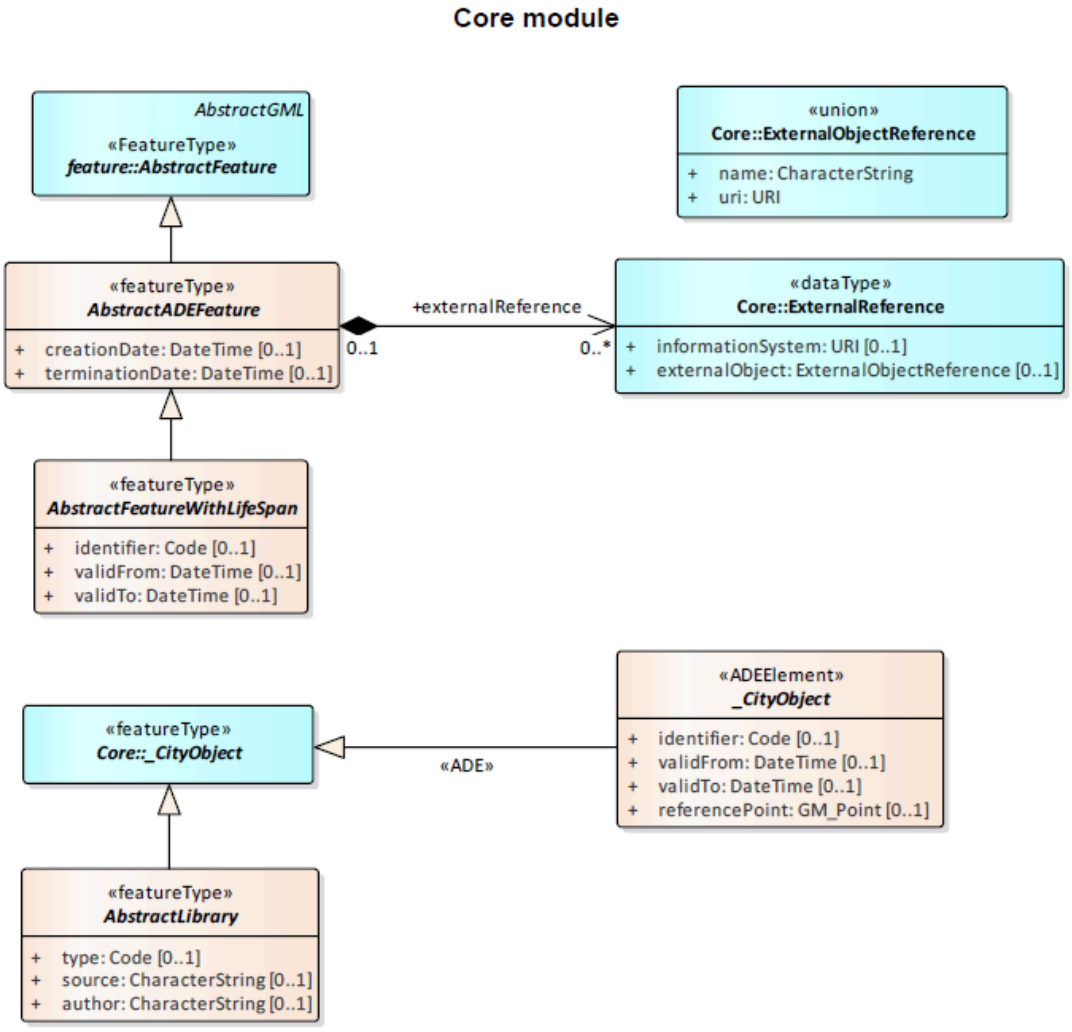


Figure 4.9: UML of the Core Modules for Scenario ADE beta 5

The `Scenario` class is kept from Beta 3 but moved off `core:_CityObject`. It now extends `AbstractFeatureWithLifeSpan`, so it picks up the lifecycle attributes but stops being a CityObject. Its attributes are a `type` code, a `projectName` and a list of `AbstractGenericParameter` entries. The `dependsOn` association from Beta 3 is removed, so the lineage handle between scenarios is gone in this iteration. The list of scenario-level modifications is also gone, since the modification classes themselves were dropped.

The `Simulation` class is new and sits next to `Scenario`. It also extends `AbstractFeatureWithLifeSpan` and carries a `type` code, a `temporalResolution` code and its own list of `AbstractGenericParameter` entries. A `Scenario` references zero or more `Simulation` instances through the `+simulation` association. The same `Simulation` can therefore be reused by multiple scenarios, since it lives in the `SimulationLibrary` rather than being composed by a single `Scenario`.

`AbstractMember` is the new shared parent of `ScenarioMember` and `SimulationMember`. It carries the `operation` attribute, which uses the same `MemberOperationValue` enumeration as Beta 3 now extended with a `replaced` value, and a list of `AbstractGenericParameter` entries. The link to the city objects targeted by a member is expressed through three property slots introduced on

`AbstractMember`. `newObjectMember` and `oldObjectMember`, each with cardinality `[0..1]`, carry the new or removed feature. `appearanceMember`, also `[0..1]`, carries the appearance counterpart. the appearance was chosen to be added to in the case the schema needs to support scenarios involving visual changes such as painting or re-cladding. This is never tested on a scenario, however appearances are features which automatically can be supported in teh nxt iteration without specifying them. All three are typed as `gml:FeaturePropertyType`, so any feature can sit in any of them.

`AbstractGenericParameter` is unchanged from Beta 3. It still carries a name, a description and a value of one of nine primitive types. What changes is where it can be attached. In Beta 5 it appears on `Scenario`, on `Simulation`, on `ScenarioMember` and on `SimulationMember`. The single member-level slot from Beta 3 is now four slots distributed across the scenario and simulation sides.

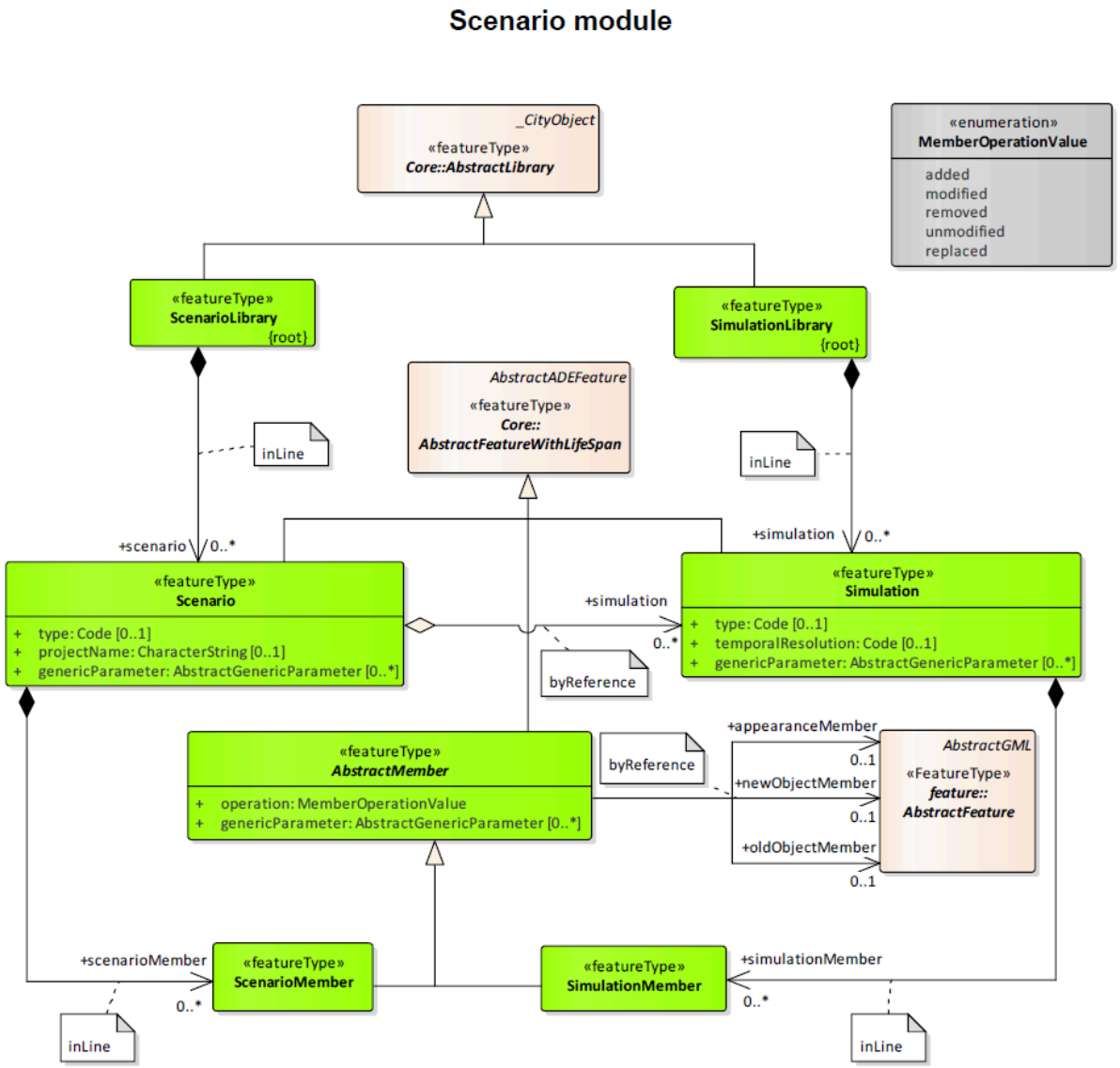


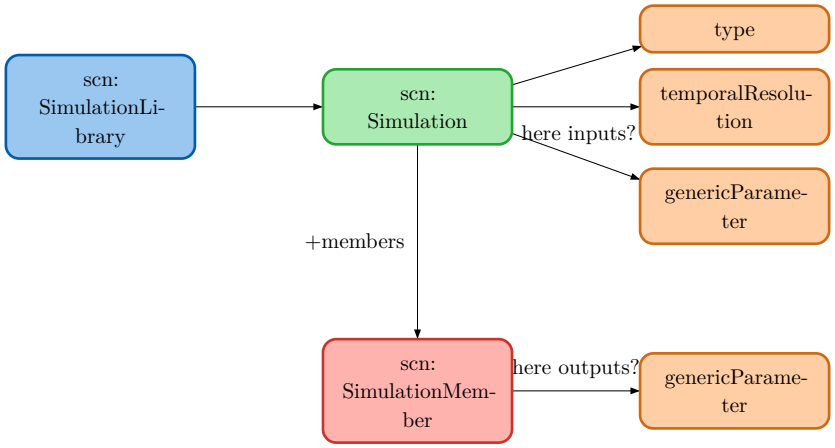
Figure 4.10: UML of the Scenario Module for Scenario ADE beta 5

4.3.2.2. Semantic Mapping

Applying the evaluation process again to the newly mapped Scenario ADE beta 5, the following mapping findings will be discussed:

- **Simulation class provides a space to map simulation parameters** *[addresses CF1 (partial); touches CF2, CF3]*
  - Mapping any of the simulation specific parameters (S1-04, S1-05, s2-06 … etc) the engine and what are input or output parameters still sit all together. `Simulation` exposes `type`, `temporalResolution`, and `genericParameter` for storing simulation-relevant parameters, and carries its city object references via `simulationMember` (not `scenarioMember`). `Scenario` links to `Simulation` via an association rather than composition, meaning multiple `Scenario` instances can reference the same `Simulation` object stored once in the `SimulationLibrary`. This is a meaningful structural improvement towards the derived structural pillars of what a scenario is. However, without structured configuration parameters to identify a specific simulation engine or what was an input ot out, it still needs some small tweaks. For example, a `DoubleParameter` named *annualHeatingDemand* is structurally identical to one named *windowToWallRatio*. The schema cannot distinguish between input and output parameters. This gap applies equally to `genericParameter` on `Simulation`.

Beta 5 Solution for Configuration: `SimulationLibrary` → `Simulation` (Attributes: `type`, `temporalResolution`, untyped `genericParameter`) → `SimulationMember` → `newObjectMember` / `oldObjectMember`

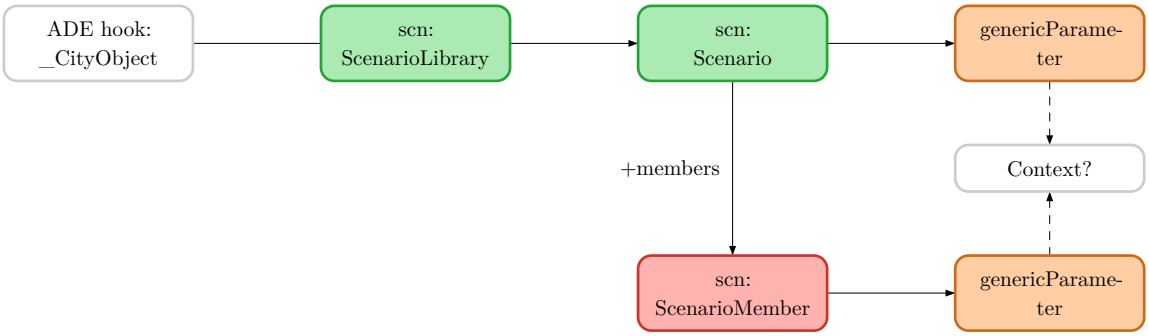


- **New modification mechanism tested for mapping strategy and result parameters** *[touches parameter S2-02]*
  - To achieve domain agnosticism, domain-specific modification types (`LayeredConstructionModification`, `ScheduleModification`) were removed entirely. Object-level changes are now expressed exclusively through `newObjectMember`, `oldObjectMember`, and `appearanceMember` properties on `AbstractMember`, all typed as `gml:FeaturePropertyType`. This shift solves domain dependency but forces entire feature geometries to be duplicated in order to adjust scalar attributes, which is inefficient. For almost all scenarios, storing the results meant having to replace the entire object it was added to instead of just appending or replacing the attribute directly from where it is stored on the object. This was accepted temporarily for isolated testing but will be addressed in the next iteration. It should be noted that the Xpath suggestions was retrospectively added to beta 3, based on discussions from the finding of this point here in beta 5. The reasoning with an encoding example is found in Section C.1.

Beta 5 Solution for Modification: `ScenarioMember` → `oldObjectMember` (existing feature) and `newObjectMember` (fully duplicated replacement feature)

- **Context parameters can now be mapped on `Scenario` and/or `Simulation`** *[addresses C1; touches parameters S1-01, S1-02, S1-03, and C3 (still unmet)]*
  - With the `SimulationLibrary` added, the `Scenario` can hold parameters that are structurally distinct from what belongs to a `Simulation`. The scenario level context now sits solely on the `Scenario` through its `genericParameter` association, without being conflated with the simulation parameters. The plausible narrative (S1-03) and the planning horizon (S1-02) which all scenarios have are stored on the `Scenario` for this reason. The weather file (S1-01) is contextual, but it is consumed by the engine, so it now finds a better place on the `Simulation` rather than sitting on the scenario or on a member as in Beta 3. Storing context on the `Scenario` through `genericParameter` is also slightly more formal than using the CityGML Generics Module directly, and it accepts links to external files or scenario reports through a `UriParameter` or `StringParameter`. The split is still by convention though. C3 remains unmet, since `AbstractGenericParameter` carries only a name and a description and cannot mark a value as input or output.

Beta 5 Solution for Context: `core:_CityObject` → `ScenarioLibrary` → (`gml:_Feature`) `Scenario` → `genericParameter` → eg: `UriParameter`



- **Result of scenario still cannot be mapped correctly** *[touches CF2, CF3; result parameters across S1–S4 remain unmapped]*
  - The framework for storing simulation output results remains still unresolved. The `genericParameter` on `Simulation` can hold untyped values, but there is no mechanism to associate results with specific city objects, specific simulation engine runs, or to distinguish between multiple result sets within the same simulation. The provenance cannot be traced.

Beta 5 Solution for Result Storage and Provenance: No clean solution exists within the Beta 5 schema without the introduction of a dedicated result class.

4.3.2.3. Validation Assessment

The bullets below report what the Beta 5 architecture revealed when the four test scenarios were compared through the schema.

- **Attribute-level modification has no schema-level home** *(R5.1) [addresses S1; touches S2–S4]*
  - This iteration tested whether the new generic member structure could still express attribute changes. It cannot. The Beta 3 `PropertyModification` class was removed to commit fully to the new `AbstractMember` design, and the `replaced` enum value was added with the intent of covering replacement cases. Practical testing revealed that no operation and member-content combination in Beta 5 means "change attribute X on feature Y". The `modified` enum value is accepted as a string but the schema has no member level slot to express which attribute was modified. The only working path is to swap full features through `oldObjectMember` and `newObjectMember`, which forces geometry duplication for what may be a single scalar change. The test outcome carries directly into Beta 6, which reintroduces a dedicated attribute modification mechanism through the `AttributeTransaction` class.
- **No lineage between scenarios** *(R5.2) [addresses Scenario Set Coherence (F2)]*
  - Beta 3′s `dependsOn` feature is removed entirely, setting aside scenario tracking and the Scenario Set Coherence intention. It is known from tests that the scenario ADE should support such lineage and inter-scenario relationships. This allows for comparison and better tracking. Beta 5 does not offer a replacement mechanism, beta phase does not have any topological relationship between any scenario. They all stand independent of each other. As mentioned this was however not the point of the beta 5 tests, so lineage will be brought back in the next iteration through the `VersionTransition` class.
- **Slot typing and operation pairing are left to the application layer** *(R5.3) [addresses S1–S4]*
  - The member slots (`newObjectMember`, `oldObjectMember`, `appearanceMember`) are typed `gml:FeaturePropertyType`, so any feature can occupy any slot. Swapping a wall for vegetation is structurally equivalent to swapping a wall for a wall. The schema also does not constrain which slots are valid for a given operation, so a file with `operation="added"` carrying only an `oldObjectMember` remains schema valid. Solutions such as using Object Constraint Language (OCL)[10] or Schematron language[11] to assert rule-based validation is out of the scope of the schema. Validation of slot type and operation pairing is therefore assigned to the application layer rather than the schema.

Code	Addresses	Recommendation
R5.1	ST1, (S2–S4)	Reintroduce a dedicated attribute-level modification path so that scalar changes do not force a full feature swap with duplicated geometry. Adopted in Beta 6 through the `AttributeTransaction` class.
R5.2	Scenario Set Coherence (F2)	Reintroduce a lineage mechanism between scenarios so dependency and ordering can be traced. Adopted in Beta 6 through the `VersionTransition` class.
R5.3	S1–S4	The member slots are `gml:FeaturePropertyType` and the operation is not bound to the slots it requires. Both are left to the scope of an application layer.

Table 9: Summary of Beta 5 validation recommendations.

[10] https://www.omg.org/spec/OCL/2.0/About-OCL
[11] https://www.schematron.com/

4.3.2.4. Insights and Transition

The status of each pillar after the Beta 5 evaluation is summarised below.

Context

- `Scenario` now carries its own `genericParameter` slot, so context parameters are better supported at the `Scenario` level rather than onto a member as in Beta 3.
- Beta 5 also explores the conceptually cleaner option by moving `Scenario` off `core:_CityObject` to `AbstractFeatureWithLifeSpan`, so a scenario represents something closer to what it is instead of a physical city object. This aligns with the `AbstractFeatureWithLifeSpan` pattern of CityGML 3.0. It comes at a cost though, since the `Scenario` loses the native `gen:*` generic-attribute hook that a `core:_CityObject` provides. Beta 6 revisits this trade-off.
- The boundary between context and strategy is still convention rather than enforcement. Both pillars use `genericParameter`, just at different levels. C3 (input vs output) is unmet because `AbstractGenericParameter` carries only a `name` and a `description`.

Strategy

- The four Beta 3 modification subclasses are dropped to break dependency on the Energy ADE and favour ADE agnosticism. The cost is that attribute-level modification loses its schema home, so a scalar change like S2-02 forces a full feature swap with the geometry duplicated (R5.1).
- The three member slots are typed `gml:FeaturePropertyType`, with no enforced constraint on slot type or on operation-slot pairings (R5.2). Parent feature referencing still need to be addressed for additions (S3-02, S4-02).

Configuration

- `Simulation` and `SimulationLibrary` give simulation runs a separable home and let multiple scenarios reuse the same `Simulation` instance. CF1 is partially met.
- Engine identifier, settings, and results all still travel through `genericParameter`. The schema cannot distinguish engine runs, separate inputs from outputs, or distinguish multiple result types on the same feature. CF2 and CF3 remain unresolved, which is what motivates `ScenarioResults`, `Settings`, and `Parameter` in Beta 6.

Transition towards Beta 6

Beta 5 results in a cleaner and more general schema, but the test scenarios cannot yet be expressed entirely. Attribute changes need to be resolved, results have no traceable location, and no lineage exists between scenarios. These limitations to the discovered in Beta 3 still need to be addressed. The Beta 5 structure also indicates a direction for resolving them. Its libraries already separate the scenario side from the simulation side, and the `oldObjectMember` and `newObjectMember` slots express a change as a delta from an old state to a new state against a base. A strategy can be described as such a set of changes applied at a transition between two states. The versioning module of CityGML 3.0 formalises this construct, as it models states, the transitions between them, and the changes that each transition carries. Backporting the module provides a structural basis for the context and strategy split, in which the strategy is represented by the transition (the controlled deliberate changes towards the scenario state we test) and the context remains as metadata on the scenario or simulation. Beta 6, therefore,

adopts despite with some minor changes and additions, the versioning module to resolve the open needs and to align the schema with CityGML 3.0 for forward compatibility.

4.3.3. Scenario ADE beta 6

From the evaluation of Beta 5 and the open needs it left, Beta 6 arises. The Beta 5 evaluation showed three requirements the schema could not yet meet. An attribute change without duplicating the parent geometry, simulation results need a way in which they can be traced, and scenarios needed a lineage relation between them. These three needs correspond to the core constructs of the CityGML 3.0 versioning module, namely a state, a transition between states, and the set of changes a transition carries. The project also placed increasing weight on forward compatibility with CityGML 3.0. Beta 6 therefore backports the versioning module, both to resolve the open needs and to align the schema with the direction of the standard.

The versioning module of CityGML 3.0 is shown below in Figure 4.11 as a reference.

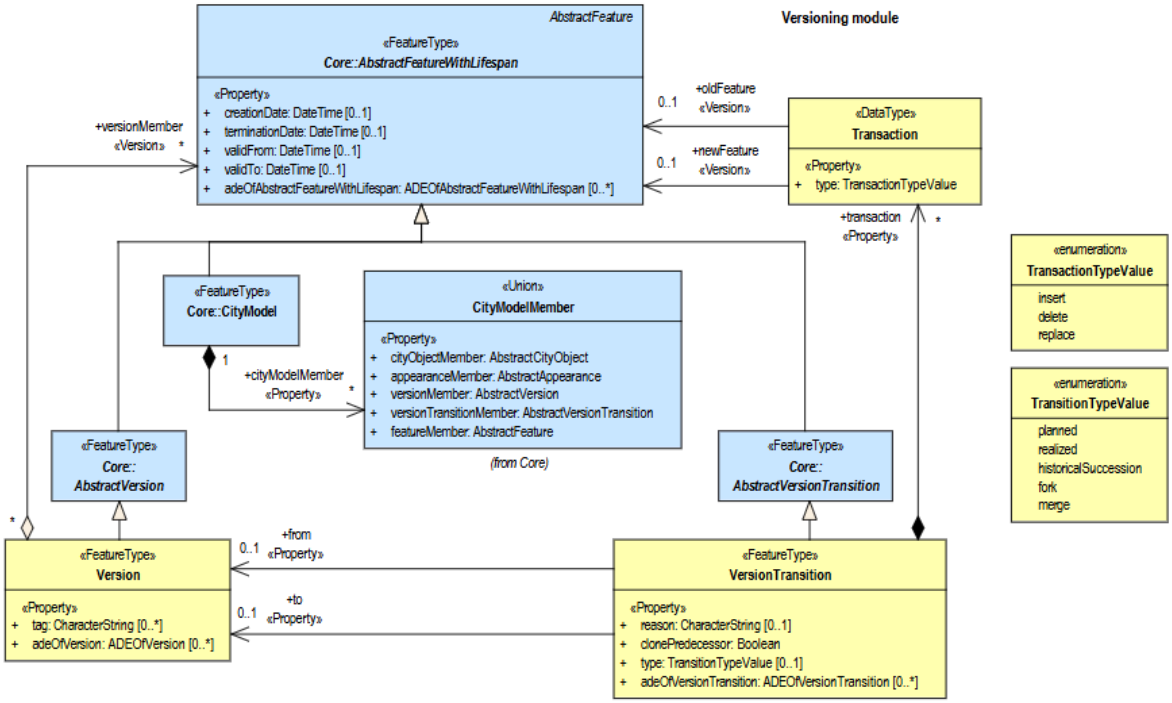


Figure 4.11: CityGML 3.0 Versioning Module

Integrating the versioning concepts from CityGML 3.0 into the Scenario ADE was a process which required mapping relevant classes and functionality from the former into the latter. The table below (Table 10) shows the alignment between the two, which forms the basis of the Scenario ADE beta 6.

CityGML 3.0 Concept	Beta 5 Equivalent	Match	Gap Beta 6 must close
`Version`	`Scenario`	Partial	`Scenario` mixes state with metadata. A `Version` keeps the state purely structural.
`VersionTransition`	None	None	Beta 5 scenarios are unrelated. Branching and chaining added through `VersionTransition`
`Transaction`	`AbstractMember` + `operation`	Partial	Beta 5 forces whole-object references. Attribute-level patching must be addressed.
`TransactionTypeValue`	`MemberOperationValueType`	Good	All actions using the `MemberOperationValueType` will be reframed in terms of the `TransactionTypeValue`
`TransitionTypeValue`	None	None	Beta 5 does not yet deal with transitions. Takover entirely.
`clonePredecessor`	None	None	Beta 5 cannot inherit unmodified objects from a prior state. A derived scenario cannot reuse members. This can be takenover which avoids duplication

Table 10: Alignment between CityGML 3.0 versioning concepts and Beta 5.

4.3.3.1. Overview beta 6

With the mapping in Table 10 as its basis, the resulting Beta 6 structure is described. Structurally, Beta 6 keeps the two module split from Beta 5 and further improves it. The core module still supplies the lifecycle and the libraries. The scenario module is reorganised around the CityGML 3.0 versioning concepts. New classes in this iteration are `Version`, `VersionTransition`, `FeatureTransaction`, `AttributeTransaction`, `ScenarioResults`, `SimulationEngine`, and the `Parameter` data type. `Settings` is renamed from `SimulationSettings`. The `AbstractMember`, `ScenarioMember`, and `SimulationMember` classes from Beta 5 are dropped, replaced by the transaction based pattern from versioning.

The lifecycle pattern first adropted in Beta 5 is preserved. `AbstractFeatureWithLifeSpan` still extends `AbstractADEFeature`, and the same lifecycle attributes are still injected into `core:_CityObject` through the "ade-hook". The `Scenario` class, however, returns to `core:_CityObject` through a new `AbstractScenario` parent, reversing the Beta 5 move to a pure feature. In CityGML 2.0 only a `core:_CityObject` subclass receives a `gml:id`, and use the core CityGML `genericAttribute` from the Generics Module. The last point can be considered a benefit, since it restores the native `gen:*` generic-attribute on the `Scenario` allowing theoretically more customiseability. The cost is that the conceptually unwanted inherited attributes flagged in Beta 3 return.What changes for `Scenario` beyond its base is what it connects to.

Instead of composing its own member list, it now references a `Version`, a `VersionTransition`, and a `ScenarioResults` block.

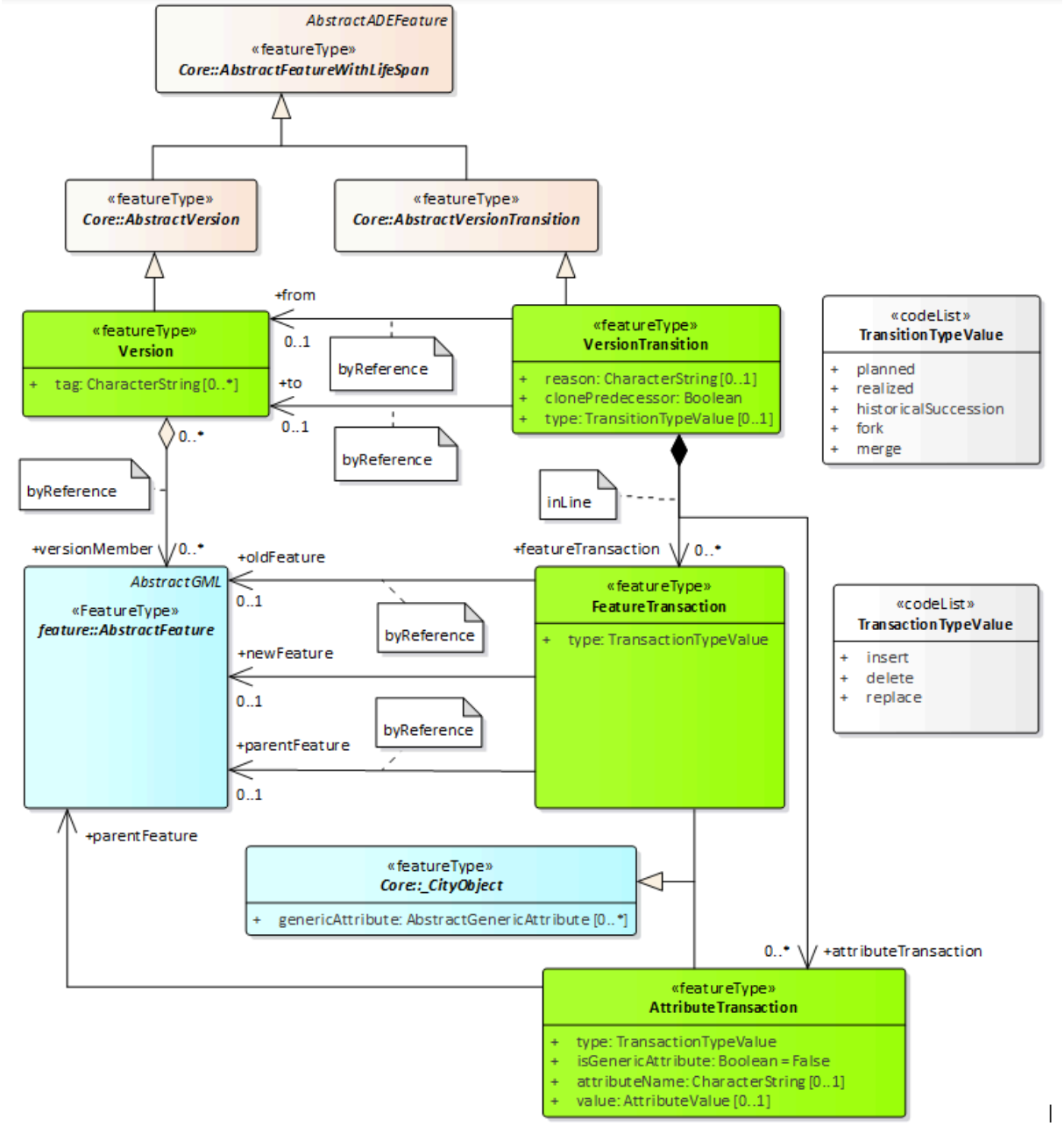


Figure 4.12: UML of the Versioning Module for Scenario ADE beta 6

`Version` is the structural snapshot of the city model at a point in time. It extends `AbstractFeatureWithLifeSpan` and carries a list of `versionMember` references typed as `gml:FeaturePropertyType`, plus an array of free `tag` strings. Each `versionMember` points to a city object that exists in that snapshot. A `Version` does not store the changes themselves, only what is present.

`VersionTransition` is the bridge between two `Version` instances. It carries `from` and `to` references, a `reason` string, a `type` code (planned, realized, and so on), a `clonePredecessor` boolean, and one or more transaction slots. When `clonePredecessor` is true, every member of the source version is implicitly carried forward to the target, and the transactions express only

the delta. This is the mechanism that solves the duplication problem flagged in the Beta 5 evaluation.

`FeatureTransaction` and `AttributeTransaction` are the two concrete delta classes. A `FeatureTransaction` carries a `type` code (insert, delete, replace), an `oldFeature` xlink, a `newFeature` (xlink or inline), and an optional `parentFeature` xlink for the previously discussed child feature storing. An `AttributeTransaction` describes a single attribute change. It carries an `attributeName` string, an `attributeType` enum (`simpleAttribute` or `cityGMLGenericAttribute` now), a typed `value`, and one or more `parentFeature` xlinks to the affected features. Allowing more than one `parentFeature` lets a single transaction cover several features that undergo the same change, so a uniform edit such as one refurbishment value applied across many buildings does not need a separate transaction per feature. Conceptually a transaction is a data type. It records the details of a change and is not a real-world feature in the city. In CityGML 3.0 it is modelled exactly that way, as a `dataType`. In this CityGML 2.0 ADE both classes instead extend `core:_CityObject`. This is a deliberate pragmatic choice stemming from FME write compatibility, and promoting a data type to a city object also happens to have practical advantages. As a `core:_CityObject`, each transaction sits inside `cityObjectMember`, so the FME CityGML reader and writer can handle it. Each one is also a top-level, addressable object with a `gml:id` for xlinking and 3DCityDB tracking, which an inline data type would not be, and it exposes the `_GenericApplicationPropertyOfCityObject` hook that `AttributeTransaction` uses to carry a `gen:*` attribute when `attributeType` is `cityGMLGenericAttribute`. A future port to CityGML 3.0 would revert the base class to a `dataType`.

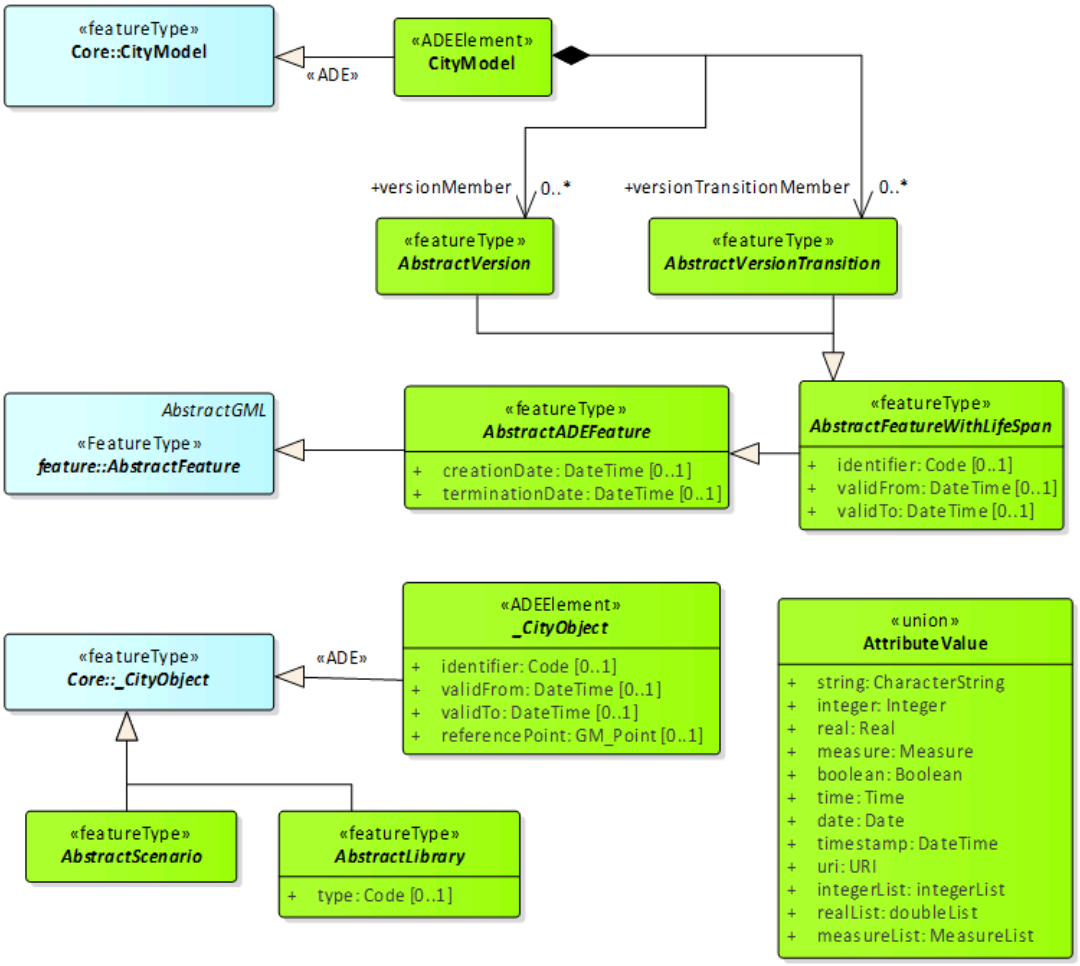


Figure 4.13: UML of the Core Module for Scenario ADE beta 6

`ScenarioResults` is the structural home for simulation outputs. It is associated with a `SimulationEngine` and a `Settings` instance from the `SimulationLibrary`, and it owns its own list of `AttributeTransaction` and `FeatureTransaction` entries. This is how a building's predicted heating demand is attached back to the building. The result lives in an `AttributeTransaction` with a `parentFeature` xlink to the building, and the `ScenarioResults` wrapping it records which engine and which settings produced it. The same pattern carries TimeSeries outputs through `FeatureTransaction` instead.

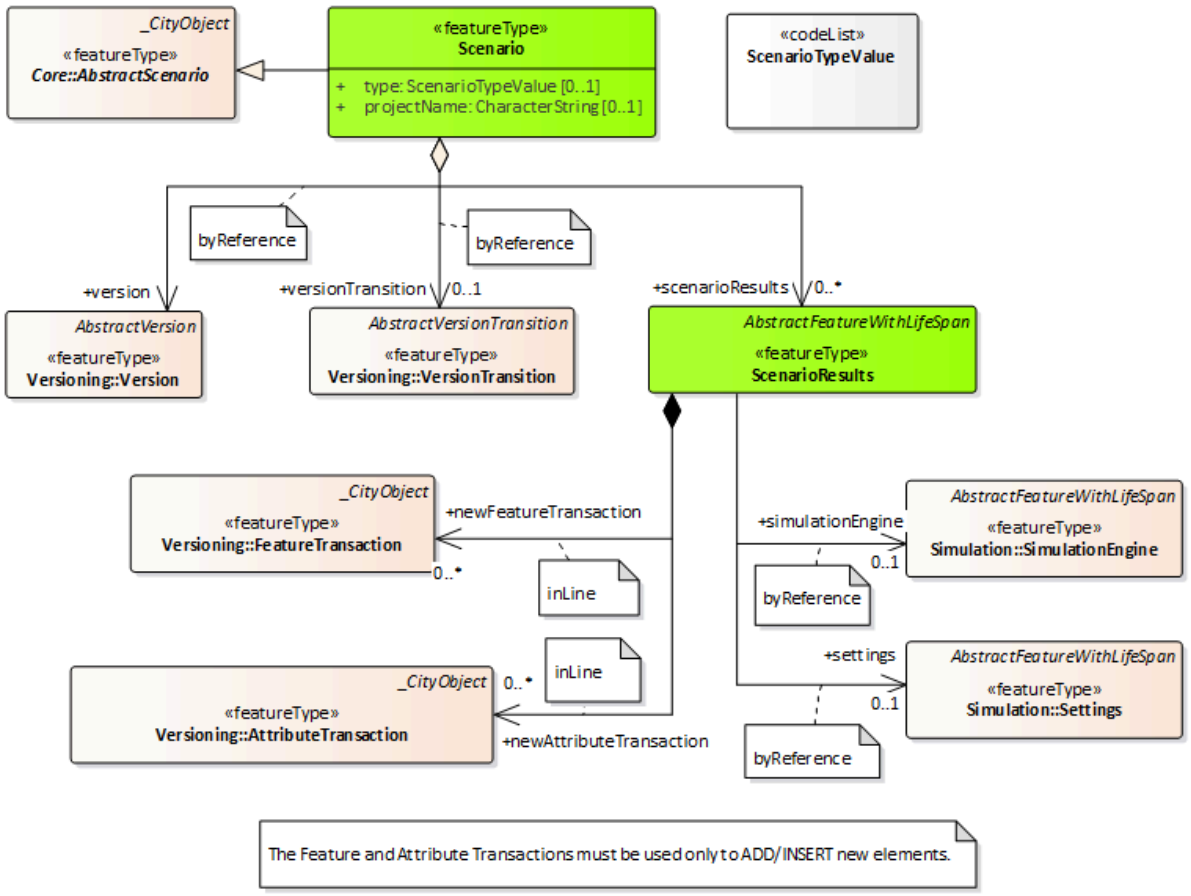


Figure 4.14: UML of the Scenario Module for Scenario ADE beta 6

The `SimulationLibrary` and `Simulation` classes from Beta 5 stay, but the simulation-specific configuration is restructured. `Settings` (renamed from `SimulationSettings`) contains one or more `Parameter` data types. Each `Parameter` carries a `name`, a typed `value`, and an optional `scope` enum (`input`, `output`, or `other`). The `scope` attribute is what allows a single `Settings` instance to be reused across scenarios while still letting downstream tools tell which parameters are simulation inputs and which are outputs.

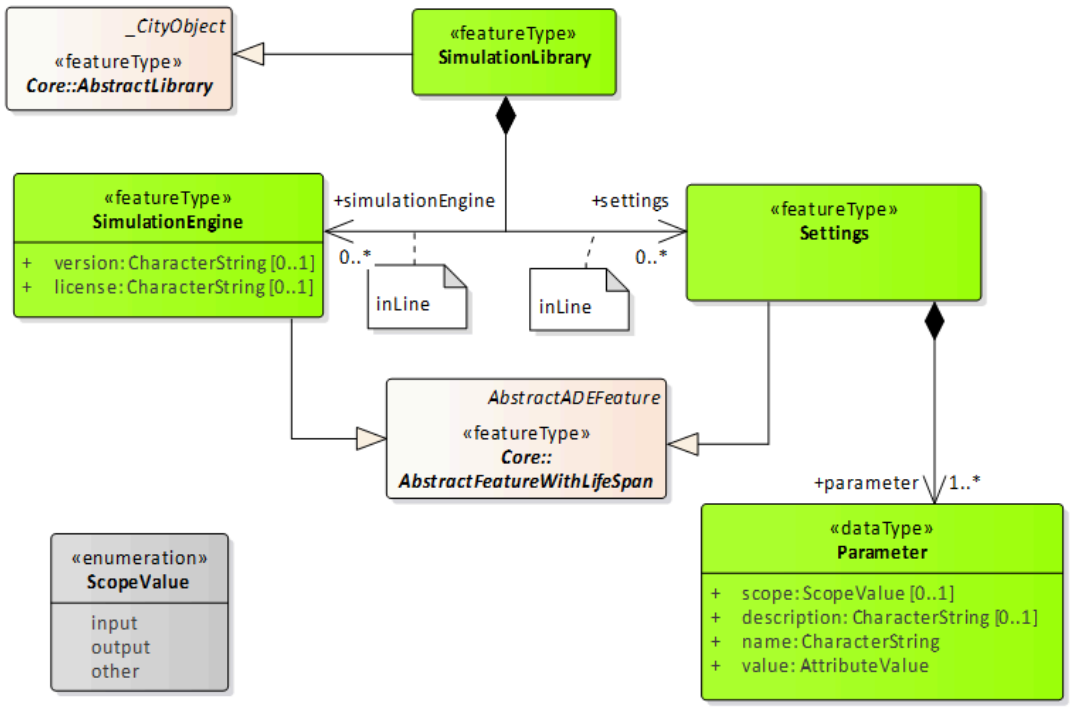


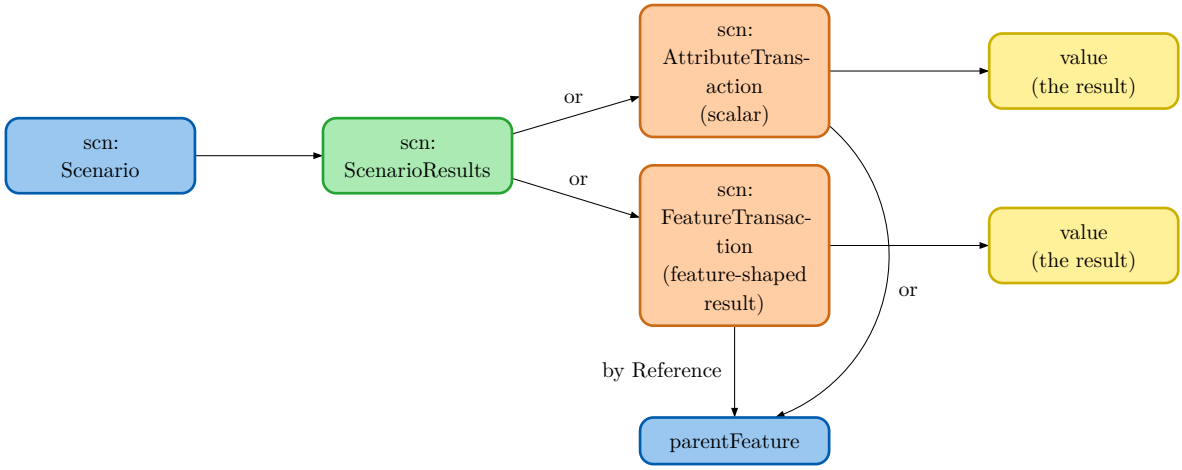
Figure 4.15: UML of the Simulation Library for Scenario ADE beta 6

4.3.3.2. Semantic Mapping

Applying the evaluation process again to the newly mapped Scenario ADE beta 6, the following mapping findings will be discussed:

- **Storage of Simulation Results** *[addresses parameters S1-06, S2-07, S3-05, S3-06, S4-06-S4-08; touches CF2, CF3]*
  - The biggest and most important change made in beta 6 is the introduction of a mechanism to deal with the simulation results and tie them to the specific citybject or feature they belong to. This is handled by the `ScenarioResults` class, which stores the combination of the `SimulationEngine`, `Settings` and the results in either an `AttributeTransaction` for scalar and list-typed values or a `FeatureTransaction` for results delivered as full features (such as a Energy ADE TimeSeries feature object). The monthly series in the test scenarios, the monthly heating and cooling demand (S1-06) and the monthly roof irradiance (S3-06), are stored as a `realList` or `measureList` in an `AttributeTransaction.value` linked to the building or surface through `parentFeature`. These re-use the transaction logic in order to be referenced to the correc cityobject or feature (creating a link between the result and the object/ feature). The mapping of the results can be shown in the mapping below:

Beta 6 - Storing Scenario Results: `Scenario` → `ScenarioResults` → (`AttributeTransaction` or `FeatureTransaction`) → `value`



- Keeping the attributes placed inside the scenario containers was a deliberate choice over hanging results directly on the original object. Given a example case of attaching a energy demand result on a building. The Energy ADE offers that through its `AbstractResource` hook on a `CityObject` (Section 4.3.1.2), as domain agnosticism the result can be stored as a CityGML `genericAttribute` on the `CityObject`. The problem is the same either way, if building takes part in several scenarios there is no way to know which result came from what scenario. On the object, the results from each scenario pile up on the same building and can no longer be told apart, and the link to the run that produced them is lost. Storing them on the `Scenario` inside `ScenarioResults`, with a `parentFeature` xlink back to the building, keeps each scenario's results separated while still tying every value to the feature that produced it. Figure 4.16 diagrammatically shows the mapping choice for why all results are stored within the Scenario ADE container. Furthermore, this is only valid for if the target/holding entity is a `CityObject`. A `Feature` or any other non-city object do not support the `genericAttribute` or can hold a `AbstractResource`.

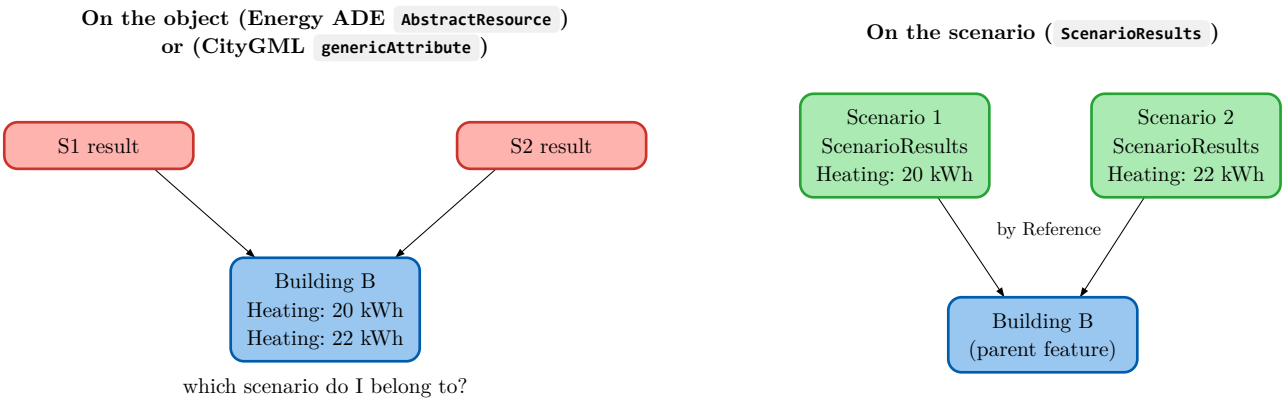
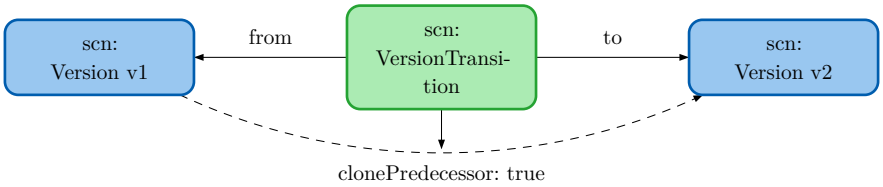


Figure 4.16: Two placements for a result when one building takes part in two scenarios. Left, results on the object do not know their scenario. Right, results on the scenario stay separated and still point back to the building.

- **How lineage is addressed** *[addresses Scenario Set Coherence; touches lineage across S1–S4]*
  - Beta 6 now addresses the beta 3 `dependsOn` by introducing scenarios as versions of a city model paired with formal transitions (backported fromCityGML 3.0 Versioning module). Using `VersionTransition` with explicit `from` and `to` tracking, it can concretely track the lineage between scenarios. This satisfies the traceability between states and therefore also Scenario Set Coherence intention. Every `Scenario` is required

to reference exactly one `Version`, so scenarios are always related to a baseline in some form. The `clonePredecessor` boolean on each transition is also required, forcing every transition to commit to whether the parent state is inherited. When `clonePredecessor` is true, the schema lets a derived version inherit the unchanged members of its source, so only the delta need be stored. If required, and as performed for the SQL encoding examples, the version can alternatively enumerate the full member set of each `Version`. This allowed for easier database inspection, and is also permitted by the schema. (Section 4.4.2.2). Because several transitions can share one source `Version`, scenarios branch from a common baseline into a tree rather than a flat list. In the four test scenarios the baseline leads to a future-climate state, and the refurbishment, PV, and greening scenarios each branch from that state. The full tree at neighbourhood scale is shown later in Figure 4.18. Below is an illustration of the mapping:



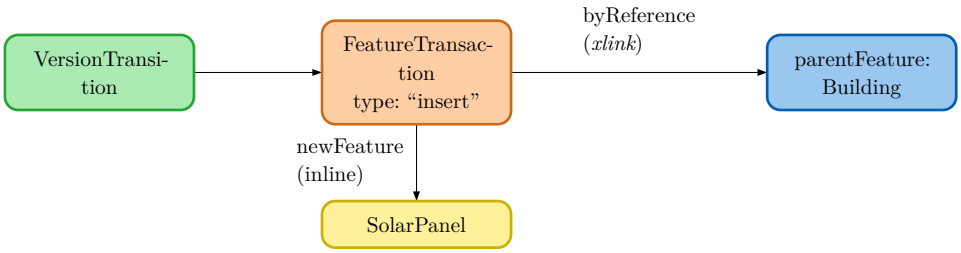
- **Resolution of the duplication problem** *[addresses parameter S2-02; touches ST1, ST2]*
  - In Beta 6 we introduce a `AttributeTransaction` class to handle targeted attribute adjustments without duplicating parent CityObject geometries. The transaction carries an `attributeName` (xsd:string), an `attributeType` discriminator, a typed `value`, and a `parentFeature` xlink to the affected feature. The discriminator is an enumeration with two values, `simpleAttribute` and `cityGMLGenericAttribute`.

> Beta 6 Solution for Attribute Modification: `VersionTransition` → `AttributeTransaction` (`type`, `attributeName`, `attributeType`, `value`, `parentFeature` xlink)

  - The `attributeType` discriminator was introduced because the data model also needs to support the insertion and deletion of new attributes on existing objects. This is what a simulation engine does when attaching results like `yearlyHeatingDemand` to buildings that never had this attribute before. A `simpleAttribute` is a typed schema attribute already defined on the parent feature. A `cityGMLGenericAttribute` carries the value as a `gen:*` child element on the transaction itself, which is then transferred to the parent feature by whatever future application is reading the schema. The typed `value` on the transaction is an `AttributeValue` union, a single choice of twelve typed alternatives. The same `AttributeValue` type is reused by `Parameter.value` in `Settings`, so settings and results carry their value through one mechanism rather than the eight separate parameter subclasses of Beta 5.
  - Three mechanisms for changing a single attribute were considered across the refinement. Option 3, the `AttributeTransaction`, was adopted into Beta 6. The `parentFeature` xlink anchors the change on the stable `gml:id` that the rest of the schema already rely on. The full comparison of the three options, the cases they cover, and their encodings is given in the appendix (Section C.1). One limitation remains for referring to `cityGMLGenericAttributes` which is a DataType, together with the weakness it keeps from the unconstrained `attributeName`. This is carried into the discussion (Section 6).

- **Addresses child features through referencing parent features** *[addresses parameters S3-02, S4-02]*
  - Dealing with features which need to be linked to a parent feature (e.g. a solar panel to a roof) is structurally supported. The `FeatureTransaction` component now houses a `parentFeature` association, connecting child objects into existing CityGML structures. We can now cleanly address requirements such as attaching new solar panels to a roof (S3-02).
  - A `FeatureTransaction` can target any feature, so the Beta 5 `appearanceMember` placeholder is no longer needed. A CityGML `Appearance` is inserted, replaced, or removed through the same `FeatureTransaction` path as any other feature.

> Beta 6 Solution for Feature Addition: `VersionTransition` → `FeatureTransaction` (type: "insert") → `parentFeature`: `Building`, `newFeature`: `SolarPanel`



- **Differentiating strategy and context** *[addresses parameters S1-01-S1-03, S2-01, S3-01, S4-01; touches C1, C2]*
  - The versioning module gives strategy and configuration a structural home but has no construct for context. A `Version` is a state, a `VersionTransition` is a change, and a `Transaction` is a delta, and none of these describe the external conditions a scenario assumes. Context is therefore the one pillar the backported module does not map onto, so it is carried as metadata across a few inherited and configuration slots. The temporal horizon can sit on the `Scenario` setting the `validFrom` and `validTo` to the timeframe the scenario occurs at or it can sit on the `Version` as `validFrom` set to the date where the scenario is set in, the narrative (S1-03) splits into two parts. The reason behind a single change is for the `VersionTransition.reason`, and the storyline of the whole scenario uses the inherited `gml:description` on the Scenario. The weather file the engine reads (S1-01) sits as a `Parameter` in `Settings`.
  - Placing the weather file in `Settings` puts it next to simulation inputs such as the radiation model or the DIN standard. That is not wrong, but the weather file is the boundary condition the strategy is tested against rather than an ordinary input. Setting its `Parameter.scope` to context, would let a reader retrieve every context parameter independently of the configuration inputs, so the context and configuration split holds structurally rather than by name alone. This suggestion is taken into the discussion.
  - The storyline narrative context currently can either sit as a `gml:description`, or in the `projectName` characterstring. Using the `gml:description` should be avoided, as it is carried by all features and which is mainly used to annotate parameters and other objects, prviding extra context. Storing the plausible scenario narrative there mixes the defining story of the scenario with ordinary free text, and it cannot be retrieved as a field of its own. A dedicated element on the `Scenario`, for instance a `narrative`

of type string, would hold the storyline explicitly and keep it apart from the generic description. This is a small addition, but it would match best what a scenario needs. This suggestion is carried into the discussion as a proposed refinement.

> Beta 6 Solution for Context:
> Weather file URI → `Settings` → `Parameter` (`scope = input`)
> Narrative description → `Scenario` → `gml:description` (inherited)
> Context Scenario horizon → `Version` or `Scenario` → `validFrom` + `validTo` (inherited from "ade-hook" on `CityObject`).

- **Improved Dealing with Simulation Settings** *[addresses parameters S1-04, S1-05, S2-05, S2-06, S3-03, S3-04, S4-04, S4-05; touches CF1, C3]*
  - Integrates `SimulationEngine` paired with the `Settings` class directly inside the `SimulationLibrary` . This was chosen to resolve configurations and allowing for the re-usability of engines to allow multiple runs to be constructed from combinations of engine and settins. This provides a structural home for the parameters which was never really there before. The `Settings` class contains one or more `Parameter` data types, each with a `name` , a typed `value` , and an optional `scope` (`input` , `output` , or `other`) that addresses C3.
  - However one limitation is that this allows for only scenarios using a simulation engine to store results or have the changes they produce be tracked back to the specific engine and settings used. The way results and their engine are combined here is one option among a few. `ScenarioResults` ties a result to the `SimulationEngine` and the `Settings` that produced it, which is in effect a small provenance record assembled from purpose made classes. The same binding could instead be expressed in the vocabulary of an established provenance standard which may increase the agnosticism of the ADE. This alternative and its trade offs are further explored in the discussion.

4.3.3.3. Validation Assessment

At this point the schema supports the four test scenarios end to end. A result can be traced to the object and the simulation run that produced it, scenarios branch from a baseline and stay comparable, an attribute can change without duplicating geometry, and a new feature records the parent it attaches to. The points below report what the evaluation still raises when the design principles are checked against the schema. The genericism inherited from backporting the versioning module raises most of them. They are ordered from the conceptual points down to the smaller item left to the application layer.

- **Scenario vocabulary is mostly inherited from versioning** *(R6.1) [addresses S1–S4]*
  - `FeatureTransaction.type` , `AttributeTransaction.type` and `VersionTransition.type` are `gml:CodeType` carried straight from the versioning module. Their labels describe versioning operations rather than scenarios, and they are kept as inherited on purpose to preserve forward compatibility with CityGML 3.0. The one field that is genuinely scenario specific is `Scenario.type` , which is given a dedicated `ScenarioTypeValue` codelist. Different values were inserted in this codelist during the encoding. The recommendation for vocabulary is presented in the discussion.
- **Typing and naming checks are left to the application layer** *(R6.2) [addresses S1–S4]*

  - Two openness points are considered out of the scope of the schema. The `versionMember` and the transaction member slots are typed `gml:FeaturePropertyType`, so any feature can occupy any slot. `AttributeTransaction.attributeName` is an unconstrained `xsd:string`, so the schema cannot tell apart two transactions that target the same `parentFeature` with the same name, nor a core attribute from a same named generic attribute. The data model should indicate where it fits but not necessarily how it is filled in, so these checks should belong to the application layer.

- **Transition branch limits** *(R6.3) [addresses Scenario Set Coherence]*
  - Each `VersionTransition` carries one `from` and one `to`, so a single transition links one parent version to one child. Merging several parents into one child is expressed as several transitions that share a target. This is the deliberate design of the CityGML 3.0 versioning module introduced by Chaturvedi et al. (2017), and it keeps the scenario graph close to a tree. Resolving the merges would be left to the application layer.

Code	Addresses	Forward Recommendation
R6.1	S1–S4	The transaction and transition `type` fields are `gml:CodeType` inherited from versioning and kept for forward compatibility. `Scenario.type` is used and the dedicated `ScenarioTypeValue` codelist recommendation is discussed in the recommendation.
R6.2	S1–S4	`versionMember` and the member slots are `gml:FeaturePropertyType`, and `attributeName` is an unconstrained `xsd:string`. These typing and naming checks are left to the application layer.
R6.3	Scenario Set Coherence	Each `VersionTransition` has one `from` / `to`. Multi-parent merges are expressed as several transitions sharing a target, the deliberate design of the CityGML 3.0 versioning module (Chaturvedi et al. (2017)). An application layer would therefore be in charge to resolve the combined state at the shared target.

Table 11: Summary of Beta 6 validation assessment and forward recommendations.

FME writer limitation

One FME writer limitation surfaced during testing. Reading scenario GML files works correctly, but writing inline `newFeature` objects (such as a PV panel inside a `FeatureTransaction`) silently drops the inline content and produces an empty `newFeature`. The files still validate against the XSD and operations expressed via xlink are unaffected. This is one of the FME limitations that motivate the move to a Python writer in Section 4.4.

4.3.3.4. Insights and Transition

The status of each pillar after the Beta 6 evaluation is summarised below.

Context

- Context is the one pillar the versioning module does not provide a construct for. A `Version` holds a state and a `VersionTransition` holds a change, and neither describes the conditions

a scenario assumes. Context is therefore carried as metadata across three inherited and configuration slots.

- The horizon can sit either on the `Scenario` or on the `Version` as `validFrom` and `validTo`, the narrative can sit in the inherited `gml:description` at the `Scenario` level, and the weather file sits as a `Parameter` in `Settings`. Placing the narrative as `gml:description` does not differentiate it enough for any other description that a parameter or feature might have. Therefore a `narrative` slot with open string should be added to the `Scenario` class. Placing the weather file in `Settings` puts it next to the simulation inputs, which is what it is, but it should be set apart as the boundary condition the strategy is tested against. These two changes suggested above are revisited in the discussion.
- `Parameter.scope` also lets a value declare whether it is an input or an output, so its role is structurally distinguished (C3).

Strategy

- `AttributeTransaction` lets a single attribute be changed without duplicating the parent geometry, which resolves the duplication problem set aside in Beta 5 (ST1).
- `FeatureTransaction` carries a `parentFeature` reference, so an added feature such as a solar panel or a tree records the feature it attaches to (PV panel). The case first raised in Beta 3 is now possible.
- Strategy lives on the `VersionTransition` as a set of transactions, and `clonePredecessor` carries unchanged members forward so the transition need only hold the delta.

Configuration

- `ScenarioResults` gives simulation outputs a home for the first time. A result is stored as an `AttributeTransaction` with a `parentFeature` link back to the building or surface, and the `ScenarioResults` records which `SimulationEngine` and `Settings` produced it. This resolves CF2 and CF3.
- `SimulationEngine` paired with `Settings` makes a simulation run reusable across scenarios, which addresses CF1.
- Typed values for both settings and results travel through one `AttributeValue` union, so a single mechanism carries every value type. The alternatives are examined in the discussion (Section 6.3.2).

Transition to the final encoding

The Scenario ADE was frozen at the Beta 6. It is the version carried into the empirical testing of Phase 4. The final evaluation of each beta's performance in regards to the foundational intentions and design principles is evaluated, with a brief concluding explanatory sentence. This is found in the conclusion section (Section 5.2.8).

4.4. Implementation of Final Encodings

Following the finalization of the Beta 6 schema, XSD, and DDL scripts are generated using the method explained in the methodology (Section 3.5).

Once obtained, data exchange scripts were required to transfer the Simstadt simulation CSV outputs into both the Scenario ADE XML constructs and the SQL database tables.

Initially, the XML data exchanges were processed on a small scale using FME. However, due to the FME bugs and limitations discussed previously, as well as the repetitive nature of

the workbench, migrating to Python became necessary. The following section introduces the foundational data exchange scripts used for both formats, followed by a detailed explanation of the specific approach taken for each encoding. This includes a breakdown of the implementation steps, such as testing foundational XML operations and conducting initial SQL tests on a small subset of 10 buildings before scaling to the entire neighborhood.

4.4.1. Data Exchange Scripts

As stated, both encodings rely on the same set of Python scripts to move data out of SimStadt and into either a Scenario ADE file or the 3DCityDB. The scripts are organised in two layers. A reader layer turns the raw SimStadt exports into clean tables, and a loader layer maps those tables onto the Scenario ADE constructs and writes them out. The reader layer is shared by both encodings, so the file and the database are always built from the same prepared data. This is what is depicted by the *cleaned dataframes* in Figure 4.17.

4.4.1.1. Reading the simulations outputs

SimStadt writes its results as CSV files with a commented metadata header followed by the per-building or per-surface rows. A separate reader is written for each workflow output. There is one for the energy demand results, one for the hourly load profiles, one for the PV potential results, and one for the surface and building U-values. Each reader detects the delimiter, parses the header to recover the run metadata such as the engine version. It then returns a dataframe keyed by the BAG identifier of the building or surface. The twelve monthly values are packed into a single space separated string at this stage so that they can later be stored directly as a time series.

Once the data is read, the two respective encodings deviated in approach and are therefore explained seperately. This is where their implementation steps are also explained.

4.4.2. Generating the XML encodings

The XML encodings begin with the creation of all the operation test files and some examples on how to encode the results. This was addressing the first phase of the methodology. Then, it moves towards encoding the entire neighborhood of Feijenoord, where each scenario is encoded in a self-contained file and afterwards combined together in one file.

4.4.2.1. Encoding the foundational operations and result samples

All operations are produced by one script. It builds a `CityModel` root, writes the `Scenario`, the baseline `Version` and the resulting `Version`, and the `VersionTransition` that links them. The change itself is then placed inside the transition. A `FeatureTransaction` carries the additions, deletions and replacements of whole features, and an `AttributeTransaction` carries the insert, delete and replace of an attribute. Added child features such as a solar panel, a tree or a new roof are built on the fly with a small set of geometry helpers and embedded inline, while the building they attach to is referenced by an xlink. The baseline city objects are copied straight from the an input file containing either one or three buildings filtered from baseline citygml file of Feijenoord Neighborhood.

The same approach produces the worked examples for storing simulation results. A second script reuses the three building file and, for each example, builds a `Scenario` with an inline `ScenarioResults` container linked to the `SimulationEngine` and `Settings`, then attaches the outputs as transactions. One file is emitted per encoding choice so the alternatives can be compared, covering the single versus multi-parent `AttributeTransaction`, the scalar demand

written as an agnostic, simple and EnergyADE attribute, and the monthly demand written as an agnostic measureList and as EnergyADE time series.

Every generated file is validated against the Beta 6 XSD as soon as it is written, ensuring it is valid. The nine resulting files are the building blocks demonstrated in the results (Figure 5.8) and the result files follow afterwards (Section 5.2.2).

4.4.2.2. Encoding Entire Scenario's in XML at Neighborhood Level

For encoding entire scenarios at scale, the workflow increases in complexity. Therefore, a high level overview of the full encoding high level workflow for the 4 case study scenario's can been seen below in Figure 4.17. This was applied at the scale of the Feijenoord Neighborhood. It is also presenting all of the input data that was used and the role of the reader layer, which prepares all the results into clean dataframes.

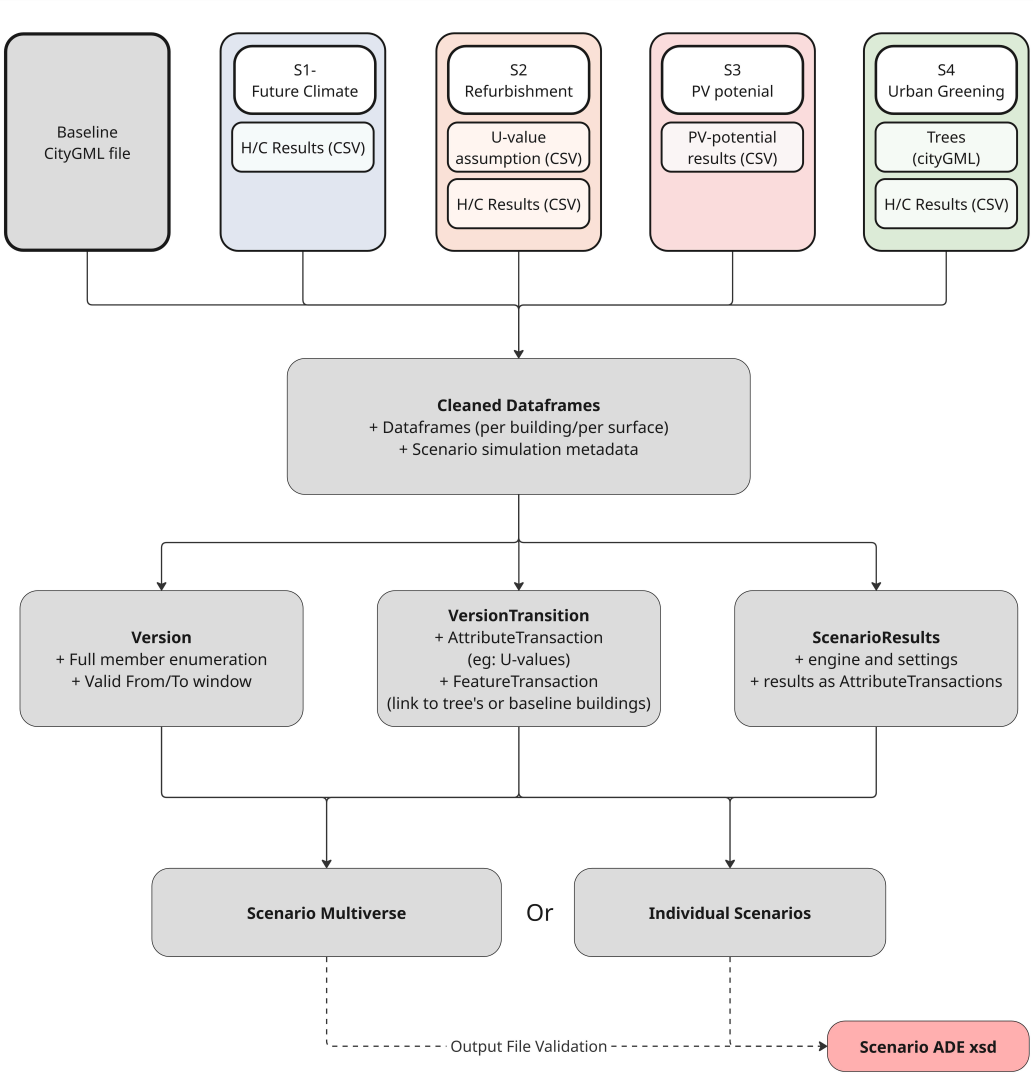


Figure 4.17: Overview of Feijenoord Scenario Creation

Where the scenario multiverse tree is build as seen below in Figure 4.18.

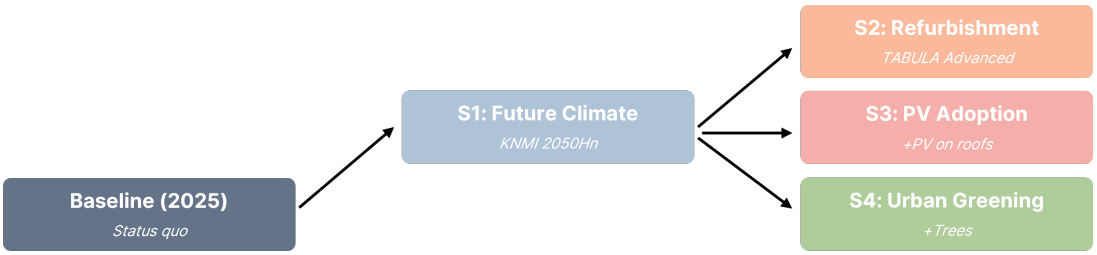


Figure 4.18: Scenario Multiverse Tree for Feijenoord

The output of the workflow are the full scenarios encoded seperately, and a combined file we call the *Scenario Multiverse*, containing the entire tree and scenario lineage within it. Both the results are found after operations and result probes, found in Section 5.2.5.

4.4.3. Generating the SQL Encodings

The SQL encodings were implemented similarly to the XML encodings. The same workflow shown in Figure 4.17 is used, with the exception that the dataframe data is loaded into the respective database tables rather than the XML constructs. To demonstrate the practical utility of the Scenario ADE and its alignment with the design principles (Table 1), the SQL encodings were produced in two steps.

First, a 3DCityDB instance is enriched with the four scenarios, containing all the same information as the XML encodings. To test and verify that the encoding was done correctly, this was first applied to a controlled sample of 10 buildings.

Enriching the database is as follows. Two reference files carrying all cityobject of each scenario (the baseline and the trees) were enriched using the 3DCityDB Importer/Exporter[12]. Then, using the BAG IDs and keys of the filtered 10-building sample, all relevant database tables were populated accordingly.

The loader layer takes the prepared dataframe and resolves every BAG identifier against the city objects that already exist in the model. This step links each result back to the exact feature that produced it, providing the traceability the schema requires. The loader then populates the Scenario ADE tables in the order dictated by their foreign keys. It begins with the simulation library, engine, and settings, continues with the version and its members, the version transition, and the scenario, and ends with the scenario results and the individual attribute and feature transactions. Every feature is also registered in the shared base table before its scenario-specific row is written.

Once the expected functionality was confirmed, the database population was scaled to include the entire Feijenoord neighborhood, mirroring the comprehensive scope of the XML scenario files.

[12] https://3dcitydb-docs.readthedocs.io/en/version-2024.0/impexp/

5. Results

This chapter presents the results of the thesis. It is split into two main parts. First, it shows the findings from the energy simulation tests. This proves the insights that guided the design principles and presents which variables test change the outcomes the most. Second, it walks through concrete examples of the Scenario ADE in action. This is presented in the format of a directory, guiding the reader to the specific encoding examples they wish to see. It shows how changes are encoded in both XML and a 3D database to provide visual evidence that the schema works as intended.

5.1. Energy Simulation Run Results

This section reports the empirical findings of the Phase 1 simulation runs described in Section 3.2 and configured in Section 4.1. Most of the runs are explorative where for each one tests whether a given input or output moves the results enough to guide what the Scenario ADE schema should on a conceptual level be able to represent. Every finding therefore closes with the schema requirement it supports, linking back to the four implications derived in Section 4.1.5 and the design principles in Table 1. One exception exists, which is the window to wall ratio comparison. This analysis is based on new data and therefore is also further evaluated, identifying how known actual values compared to standard methods using standardized archetype values affect simulations. Outliers are also inspected to understand which buildings are most affected by the change.

The findings are presented in four parts. The radiation model comparison tests a configuration input. The window to wall ratio comparison tests a building attribute input. The outlier analysis examines how unevenly that input affects individual buildings. The inhabitant comparison checks the credibility of a derived output against an external source. The section then ends with a brief comparison of the four what-if runs that are carried forward as the test scenarios of Phase 2.

5.1.1. Radiation Model Sensitivity (Perez vs Hay)

A baseline run was produced for each neighbourhood, and the radiation model was then switched from Perez to Hay while every other input was held fixed. Figure 5.1 and Figure 5.2 show how sensitive the baseline results are to this single choice.

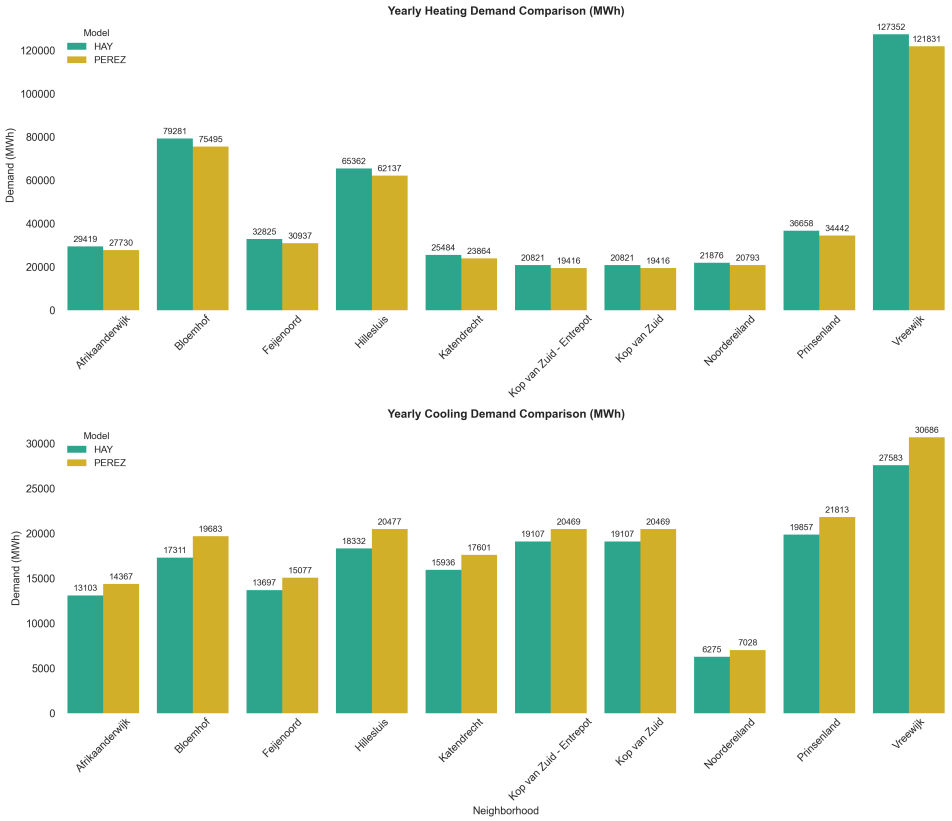


Figure 5.1: Absolute Comparison PEREZ vs HAY

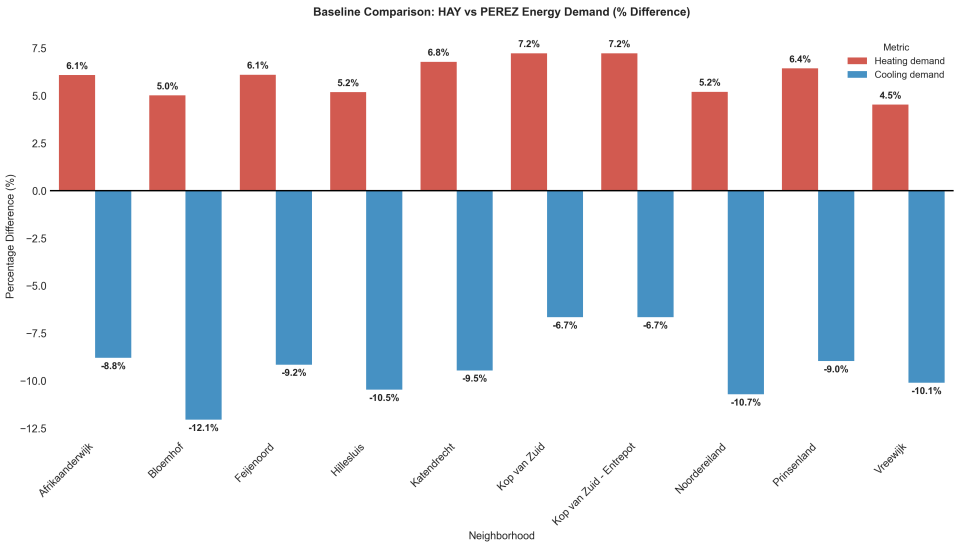


Figure 5.2: Relative Comparison Perez vs Hay

The largest change is in the cooling demand across almost all neighbourhoods, where Perez outputs a greater cooling demand in MWh than Hay. Hay in turn outputs a higher heating demand than Perez. This is consistent with how each radiation model represents incoming solar energy and how SimStadt uses it.

The Perez workflow includes mutual reflections and detailed sky brightening, so it computes a higher solar heat gain. In summer this extra energy must be offset by cooling, which produces the consistently higher cooling demand. The Hay model does not capture reflected energy and

treats the diffuse sky component more conservatively, so it predicts less free solar heating and raises the calculated heating demand to maintain thermal comfort.

The difference is not a numerical artefact but a fundamental difference in how each model represents the urban environment.

Implication for the schema: A single configuration choice belonging to a simulation such as the radiation model, shifts both heating and cooling demand. If this simulation settings are not recorded, two scenarios built from the same city model can differ for reasons that cannot be reconstructed afterwards. The schema must therefore store the parameters of each simulation run and link them to the scenario they belong to (implication 1 in Section 4.1.5, principle CF1).

5.1.2. Window to Wall Ratio: Measured vs Standardized Archetype Values

The WWR is normally inherited from a archetype rather than measured. The acquired Gemeente Rotterdam dataset makes a direct comparison possible. The measured AI values are compared against the commonly used 20% archetype, and against a value approximating the measured neighbourhood mean (7% for Feijenoord, 10% for Prinsenland). The results are shown in Figure 5.3 and Figure 5.4, with the numbers in Table 12 and Table 13.

The cooling demand is by far the most affected. The 20% uniform assumption increases cooling demand by +57.7% for Feijenoord and +61.8% for Prinsenland relative to the measured input, while the archetype close to the mean changes it by only −7.65% (Feijenoord) and −0.49% (Prinsenland). Heating demand is far less sensitive than cooling. An assumed neighbourhood-wide WWR can move the cooling demand by more than half, while a value close to the true mean stays close to the measured result.

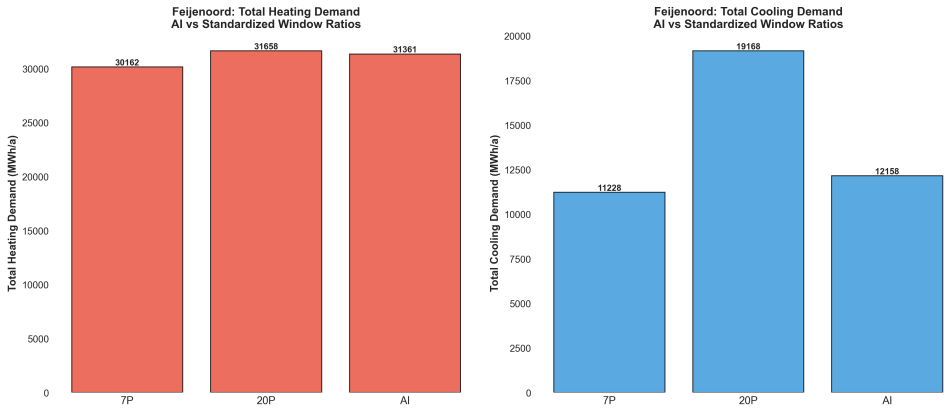


Figure 5.3: Barchart of standardized values to window wall ratio's for Feijenoord

Scenario	Heating (MWh)	Diff % to AI	Cooling (MWh)	Diff % to AI
7P	30162.04	−3.82%	11227.94	−7.65%
20P	31658.05	+0.95%	19167.88	+57.66%
AI	31360.97	-	12158.02	-

Table 12: Comparison of standardized values to window wall ratio's Feijenoord

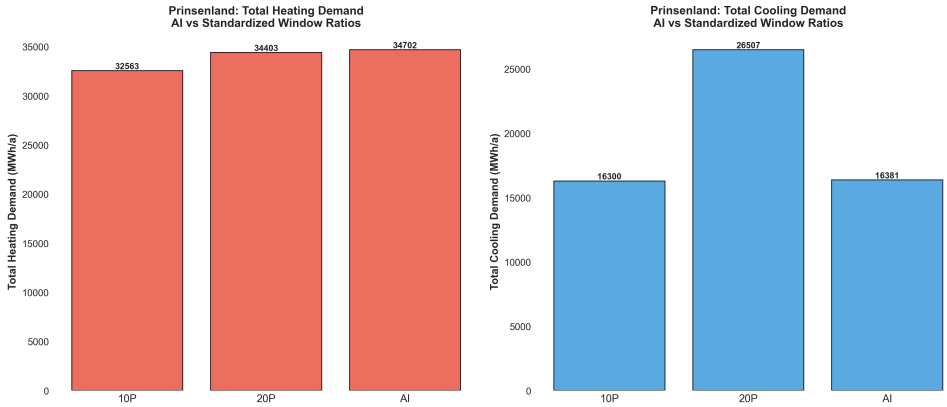


Figure 5.4: Barchart of standardized values to window wall ratio's for Prinsenland

Scenario	Heating (MWh)	Diff % to AI	Cooling (MWh)	Diff % to AI
10P	32562.65	−6.16%	16300.28	−0.49%
20P	34403.42	−0.86%	26506.66	+61.81%
AI	34701.72	+0.00%	16381.01	+0.00%

Table 13: Comparison of standardized values to window wall ratio's Prinsenland

Implication for the schema: The WWR is a property of an individual building, not of the neighbourhood, and replacing it with a uniform assumption changes the result substantially. When a scenario enriches the city model with measured data of this kind, that data has to be stored explicitly and tied to the exact buildings it describes, otherwise the result cannot be told apart from one produced under a default archetype. The schema must therefore store new attribute-level enrichment and link it back to the specific feature it modifies (implication 2 in Section 4.1.5, principles C3 and ST2).

5.1.3. Identifying Outliers from Simulation Results

This analysis asks how evenly the effect of the previous results presented is spread across individual buildings. For every building the annual demand under a standardised WWR is compared against the measured baseline, and the relative change is flagged as an outlier using the interquartile-range rule described in Section 4.1. Figure 5.5 shows the resulting spread for Feijenoord under the 20% archetype.

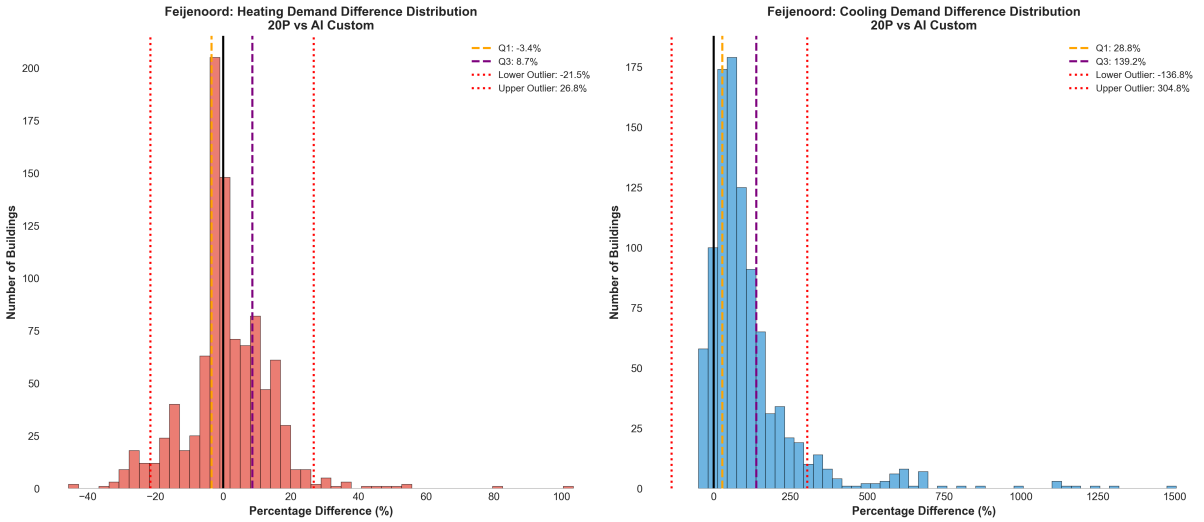


Figure 5.5: Per-building demand difference distribution for Feijenoord (20% archetype against measured AI). Left: heating. Right: cooling. The dashed lines mark the quartiles and the interquartile-range outlier bounds.

The distribution is highly uneven where most buildings sit close to the neighbourhood level change, but the cooling demand carries a long right tail that reaches well beyond +1000%, far past the upper outlier bound. The heating side is tighter and roughly centred on zero. The same shape appears in the other configurations and in Prinsenland, which are given in Section C.2.

Inspecting the flagged buildings reveals a consistent pattern. Nearly all of the largest cooling outliers are small residential building parts with very low absolute cooling demand. Because the denominator is small, adding glazing produces enormous percentage changes while the absolute change stays a fraction of a MWh. The most extreme case reaches +1505.3% in Feijenoord, a residential unit from 2016, and +701.4% in Prinsenland, a residential unit from 2006. The heating outliers behave differently. They are a mix of whole buildings and building parts with lower percentage differences, and the most extreme are larger non-residential buildings such as the −64.6% industrial building in Feijenoord and the −56.4% production building in Prinsenland. Figure 5.6 shows the most extreme cases.



Figure 5.6: The most extreme WWR outlier buildings, highlighted in red. Top row, cooling outliers: a Feijenoord residential unit (+1505.3%) and two Prinsenland residential units (+701.4%, +685.5%). Bottom row, heating outliers: a Feijenoord industrial building (-64.6%), a Prinsenland production building (-56.4%), and a Feijenoord production building (+103.5%). Cooling outliers are small low-demand units, while heating outliers are larger non-residential buildings. The full per-building tables and the matching 3D model views are in Section C.2.

Implication for the schema: The same input change lands very differently on individual buildings, and explaining any single value means going back to the exact building, its attribute state, and the run that produced it. A neighbourhood total hides this completely. The schema must therefore link every result to the specific feature, the city-model version, and the simulation run that produced it, so anomalies can be better traced and explained without re-running the simulation (implication 3 in Section 4.1.5, principle CF2).

5.1.4. The Four What-If Runs

The final group combines them into four what-if situations that each branch from the same baseline city model. They change the weather file to a future-climate context (S1), add a refurbishment strategy under that climate (S2), add rooftop PV (S3), and add street trees (S4). Figure 5.7 summarises the neighbourhood-total change against the baseline for Feijenoord, for the runs that affect building demand.

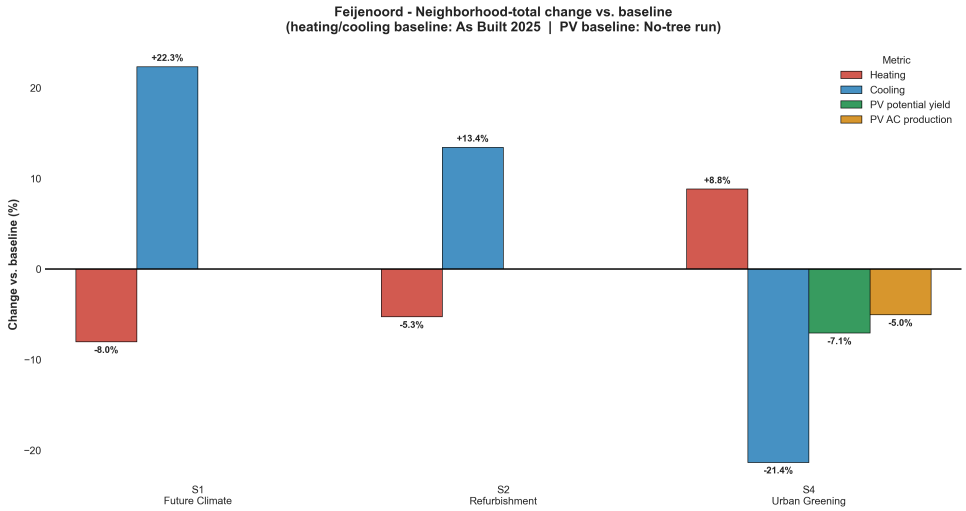


Figure 5.7: Neighbourhood-total change against the baseline for the Feijenoord what-if runs. Heating and cooling are shown relative to the as-built 2025 baseline, and the PV columns under urban greening are relative to the no-tree run.

Each run moves the demand in a different and explainable direction. The future-climate context raises cooling demand by +22.3% and lowers heating by −8.0%, reflecting the shift in temperature the KNMI scenario encodes. Refurbishment under that same climate reduces the effect, with cooling at +13.4% and heating at −5.3% against the baseline. Urban greening inverts the cooling response, lowering cooling by −21.4% through shading while raising heating by +8.8% as the trees also block winter solar gain. That same shading reduces the rooftop PV potential by −7.1%. PV adoption (S3) is not shown here as it was the benchmark PV measurement on which Urban Greening (S4) is compared to.

The precise results and values are trivial, the importance lies in their relative differences. Four alternatives, built from one shared baseline, produce results that are only meaningful when compared against each other and against that baseline. This is the comparison the schema must support, and when scenario generation is most useful. It is also why these four runs are carried forward unchanged as the test scenarios S1 to S4 that evaluate the Scenario ADE from Section 4.2 onward.

5.2. Scenario ADE Results

This section reads as a directory of worked encoding examples. Each one shows how a single kind of scenario change is represented in the Scenario ADE, and which design principle (Table 1) it puts to the test. In case curious, a reader is welcome to follow the results sequentially, if not, the Table 14 lists what each example demonstrates and where to find it, first as XML encodings and then using SQL. The performance of the Scenario ADE in regards to the intial intentions and principles can be found at the end.

To see how the Scenario ADE...	See	Page	Principle
XML Encodings (Section 5.2.1 onward)			
Operation 1: adds a new building (root feature)	Section 5.2.1.1	p.77	–
Operation 2: removes a building	Section 5.2.1.2	p.79	–
Operation 3: replaces a root feature (a tree)	Section 5.2.1.3	p.81	–

To see how the Scenario ADE...	See	Page	Principle
Operation 4: adds a child feature (solar panel on a roof)	Section 5.2.1.4	p.82	–
Operation 5: deletes a child feature	Section 5.2.1.5	p.84	–
Operation 6: replaces a child feature (a roof)	Section 5.2.1.6	p.85	–
Operation 7: inserts a new attribute on a feature (wall U-value)	Section 5.2.1.7	p.87	–
Operation 8: deletes an attribute	Section 5.2.1.8	p.88	–
Operation 9: changes an attribute value (wall WWR)	Section 5.2.1.9	p.89	–
Result Storage 1: stores a simulation result against the object and run that produced it	Section 5.2.2	p.91	–
Result Storage 2: stores a result as domain agnostic and with Energy ADE	Section 5.2.3	p.92	–
Result Storage 3: transactions carrying 1 to many parents to avoid duplication	Section 5.2.4	p.95	–
S1: Feijenoord Future Climate Scenario Encoding	Section 5.2.5.1	p.97	ALL
S2: Feijenoord Refurbishment Scenario Encoding	Section 5.2.5.2	p.100	ALL
S3: Feijenoord PV Adoption Scenario Encoding	Section 5.2.5.3	p.103	ALL
S4: Feijenoord Urban Greening Scenario Encoding	Section 5.2.5.4	p.106	ALL
Feijenoord Scenario Multiverse Encoding	Section 5.2.5.5	p.110	ALL
Individual Vs Multiverse File Size Comparison	Section 5.2.6	p.111	–
SQL Encodings (Section 5.2.7)			
Reusability of Settings and Engine	Section 5.2.7.2	p.113	CF1
Result Traceability and Provenance	Section 5.2.7.3	p.113	CF2
Results are Distinguishable	Section 5.2.7.4	p.114	CF3
Forward and Backward Scenario Lineage	Section 5.2.7.5	p.115	F1-F2
Context and Strategy Differentiation	Section 5.2.7.6	p.115	C1
Scenario Narrative and Temporal Horizon	Section 5.2.7.7	p.117	C2
Simulation Input and Output Differentiation	Section 5.2.7.8	p.118	C3
Strategy Changes Point To Where They Belong	Section 5.2.7.9	p.118	ST1, ST2
Comparing Cooling Results Across Scenarios	Section 5.2.7.10	p.119	F2
Final Conclusion Data Model			
Are the Design Principles Met?	Section 5.2.8	p.120	all

Table 14: Reader's guide to the Scenario ADE examples. Each row links an example to what it demonstrates and the design principle (Table 1) it tests.

5.2.1. XML Encoding - The 9 Foundational Operations

As introduced in the methodology (Section 3.5), the 9 foundational operations were tested. The full set is shown in Figure 5.8, and the results with explanations are presented in the following sections.

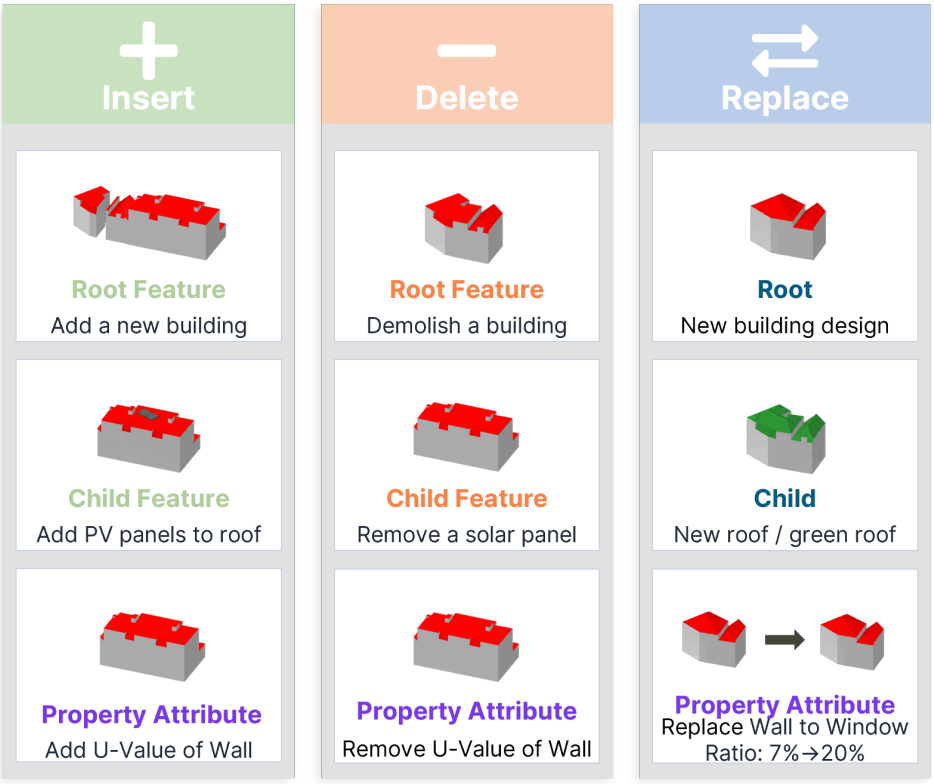


Figure 5.8: The nine foundational operations used as building blocks for the test scenarios.

> **Note**
>
> All examples shown below in the following operation sections stem from valid CityGML files, filtered from case study buildings located in Feijenoord, Rotterdam. To conserve space, some simplifications to the XML code examples are made, the complete sample files will be available in the thesis github repository. Lastly, annotations in the XML encoding examples will sometimes appear misaligned at page breaks. This is a limitation of the underlying library used to display the code.

5.2.1.1. Operation 1: Adding a root feature (new building)

The first operation tests the sole ability of the Scenario ADE to add a new root feature to the CityGML model. In this case, we add a new building to the existing block of buildings. The operation can be visualised in Figure 5.9.

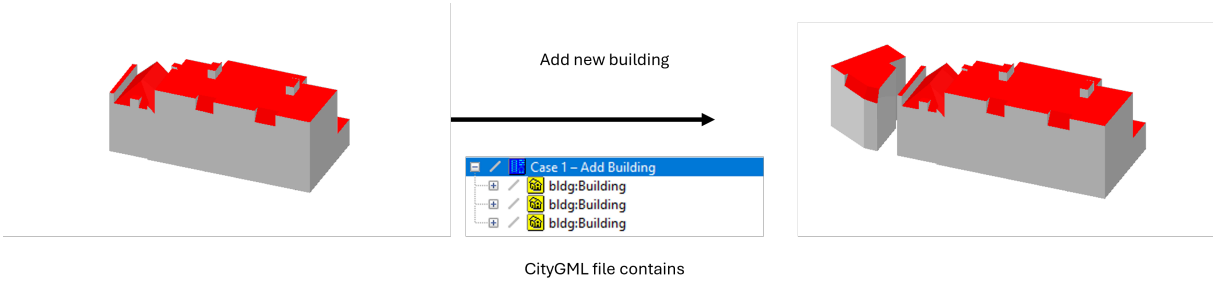


Figure 5.9: Operation 1 - Inserting a new root element (building)

The XML encoding begins by defining the parent container, the `Scenario` object. This object holds the general scenario metadata (including but not limited to the `description` (inherited from gml:AbstractGMLtype), `identifier`, `type` and `projectname`), and additionally specifying the valid lifespan and optional `gml:name`, and establishes the formal links to the resulting `scn:version` and the `scn:versionTransition` which details how this result is achieved.

The baseline model is captured in `Version v1`, referencing the two existing buildings (`BAG_...8982` and `BAG_...4925`). To define the addition of a new building, the `VersionTransition` acts as the bridge connecting `from` `v1` `to` `v2`. Within this transition, a `FeatureTransaction` of type `insert` carries the newly inserted building (`BAG_...4922`) inline inside its `scn:newFeature`. The building is packed inside the transaction rather than placed at root level and referenced by an xlink. This keeps a feature that the scenario introduces out of the baseline city model, and it mirrors how child features that cannot stand on their own are handled later in Section 5.2.1.4. The `clonePredecessor` flag is set to `true`, formally indicating that all unmodified elements from `v1` automatically carry over into the new branch.

Finally, `Version v2` serves as the complete snapshot of the scenario's resulting state. While the transition handles the isolated delta (the insertion), `v2` explicitly lists all valid members for that timeline. These are the two original inherited buildings and the newly inserted third building, which `v2` reaches by xlink, living inline inside the transaction.

Below is an example of the XML encoding of the first operation (Listing 1), with each respective classes annotated.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <gml:name>Case 1 – Add Building</gml:name>
  <gml:boundedBy>
    <!-- [envelope] -->
  </gml:boundedBy>
  <core:cityObjectMember>
    <scn:Scenario gml:id="id_scenario_case1">
      <gml:description>New building inserted into existing block</gml:description>
      <scn:identifier>id_scenario_case1</scn:identifier>
      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
      <scn:validTo>2030-12-31T23:59:59</scn:validTo>
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">predictive</scn:type>
      <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
      <scn:version xlink:href="#id_version_case1_v2"/>
      <scn:versionTransition xlink:href="#id_vt_case1"/>
    </scn:Scenario>
  </core:cityObjectMember>
  <gml:featureMember>
    <scn:VersionTransition gml:id="id_vt_case1">
      <gml:description>Building inserted</gml:description>
      <scn:reason>Permit approved, construction finished</scn:reason>
      <scn:clonePredecessor>true</scn:clonePredecessor>
      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</scn:type>
```

(1) Scenario

```
25      <scn:from xlink:href="#id_version_case1_v1"/>
26      <scn:to xlink:href="#id_version_case1_v2"/>
27      <scn:transaction>
28        <scn:FeatureTransaction gml:id="id_ft_case1_insert">
29          <gml:description>Insert new building</gml:description>
30          <scn:type codeSpace="http://.../codelists/FeatureTransactionType.xml">insert</scn:type>
31          <scn:newFeature>
32            <bldg:Building gml:id="BAG_0599100000654922">
33            <!-- Building attributes, address and geometries omitted -->
34            </bldg:Building>
35          </scn:newFeature>
36        </scn:FeatureTransaction>
37      </scn:transaction>
38    </scn:VersionTransition>
39  </gml:featureMember>
40  <gml:featureMember>
41    <scn:Version gml:id="id_version_case1_v1">
42      <gml:description>Baseline</gml:description>
43      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
44      <scn:versionMember xlink:href="#BAG_0599100000618982"/>
45      <scn:versionMember xlink:href="#BAG_0599100000654925"/>
46    </scn:Version>
47  </gml:featureMember>
48  <gml:featureMember>
49    <scn:Version gml:id="id_version_case1_v2">
50      <gml:description>Scenario after building insertion</gml:description>
51      <scn:validFrom>2027-06-01T00:00:00</scn:validFrom>
52      <!-- 618982 and 654925 carried forward from v1 by clonePredecessor -->
53      <scn:versionMember xlink:href="#BAG_0599100000654922"/>
54    </scn:Version>
55  </gml:featureMember>
56  <!-- the 2 baseline Buildings (BAG_0599100000618982, BAG_0599100000654925) sit at root; the inserted Building
    (BAG_0599100000654922) is carried inline above -->
```

(2) Version Transition — lines 25–27
(3) Feature Transaction — lines 28–36
(4) Version Transition — line 38
(5) Version v1 (baseline) — lines 41–46
(6) Version v2 (Added building) — lines 49–54

Listing 1: Adding a root feature (new building)

5.2.1.2. Operation 2: Removing a root feature (building)

In the second operation we remove a building. In this example we go from three buildings to two, as can be seen in Figure 5.10. The file contains all three buildings, where the scenario ADE handles the instruvtions and changes required to transition between versions.

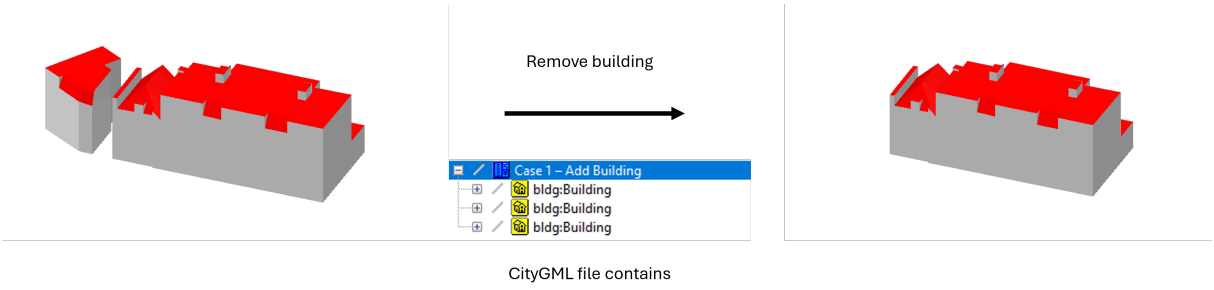


Figure 5.10: Operation 2 - Deleting a root feature (building)

The XML is very similar to the first operation, however now instead a building is removed.

Reading down the example, first the `Scenario` object is defined holding all scenario metadata, and establishing the link that results into the `scn:version` and the `scn:verstionTransition`. In this case,we specify the `Scenario` with the description *One building demolished, two remain* and assign it the `scn:type` of *explorative*.

The baseline and its references to the respective features and city objects all remain the same. The transaction type in the `FeatureTransaction` is now set to `delete`. Here the `clonePredecessor` flag is set to

`true`. The two surviving buildings carry forward from the previous version on their own. The transition only needs to record what changed. The `delete` transaction points to the demolished building through its `oldFeature` reference.

Because the members carry forward, the new version `v2` lists no `versionMember` of its own. The two unchanged buildings come from `v1` and the third is removed by the transaction. The scenario still represents a complete picture. All three buildings are still present in the file, and we use `from` and `to` to specify the version transition from the previous version containing three buildings to the final version containing only two buildings.

Listing 2 shows an example of the XML encoding of the operation.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <gml:name>Case 2 – Remove Building</gml:name>
  <gml:boundedBy>
    <!-- [envelope] -->
  </gml:boundedBy>
  <core:cityObjectMember>
    <scn:Scenario gml:id="id_scenario_case2">
      <gml:description>One building demolished, two remain</gml:description>
      <scn:identifier>id_scenario_case2</scn:identifier>
      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
      <scn:validTo>2030-12-31T23:59:59</scn:validTo>
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">explorative</scn:type>
      <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
      <scn:version xlink:href="#id_version_case2_v2"/>
      <scn:versionTransition xlink:href="#id_vt_case2"/>
    </scn:Scenario>
  </core:cityObjectMember>
  <gml:featureMember>
      <scn:VersionTransition gml:id="id_vt_case2">
        <gml:description>Building demolished</gml:description>
        <scn:reason>Structural failure</scn:reason>
        <scn:clonePredecessor>true</scn:clonePredecessor>
        <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</scn:type>
        <scn:from xlink:href="#id_version_case2_v1"/>
        <scn:to xlink:href="#id_version_case2_v2"/>
        <scn:featureTransaction>
          <scn:FeatureTransaction gml:id="id_ft_case2_delete">
            <gml:description>Delete building</gml:description>
            <scn:type codeSpace="http://.../codelists/FeatureTransactionType.xml">delete</
            scn:type>
            <scn:oldFeature xlink:href="#BAG_0599100000654926"/>
          </scn:FeatureTransaction>
        </scn:featureTransaction>
      </scn:VersionTransition>
  </gml:featureMember>
  <gml:featureMember>
    <scn:Version gml:id="id_version_case2_v1">
      <gml:description>3 buildings, before demolition</gml:description>
      <gml:name>Baseline – 3 buildings</gml:name>
      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
      <scn:tag>baseline</scn:tag>
      <scn:versionMember xlink:href="#BAG_0599100000618982"/>
      <scn:versionMember xlink:href="#BAG_0599100000654926"/>
      <scn:versionMember xlink:href="#BAG_0599100000654925"/>
    </scn:Version>
  </gml:featureMember>
  <gml:featureMember>
    <scn:Version gml:id="id_version_case2_v2">
      <gml:description>2 buildings after demolition</gml:description>
      <scn:validFrom>2027-03-01T00:00:00</scn:validFrom>
      <scn:tag>scenario</scn:tag>
      <!-- no versionMember: 618982 and 654925 are cloned from v1, 654926 is removed by the
      delete transaction -->
    </scn:Version>
```

(1) Scenario
(2) Version Transition
(3) Feature Transaction
(4) Version Transition
(5) Version v1 (baseline)
(6) Version v2 (Removed Building)

```
54      </gml:featureMember>
55  <!-- the 3 Buildings (BAG_0599100000618982, BAG_0599100000654925, BAG_0599100000654922)] -->
```

Listing 2: Deleting a Root Feature (Building)

5.2.1.3. Operation 3: Replacing a root feature (a tree)

To introduce more variety in the operation examples, instead of a building, a tree is replaced. As tree's grow they may need to be replaced in the CityModel, Figure 5.11 shows the replacement of a oak tree with it's more mature, grown version.

In order to perform such an operation, in line with the previous examples, all four tree's are present, where it is the Scenario ADE holding the instructions to transition from the previous version of the `CityModel` containing the young tree to the new version containing the new older replacement tree.

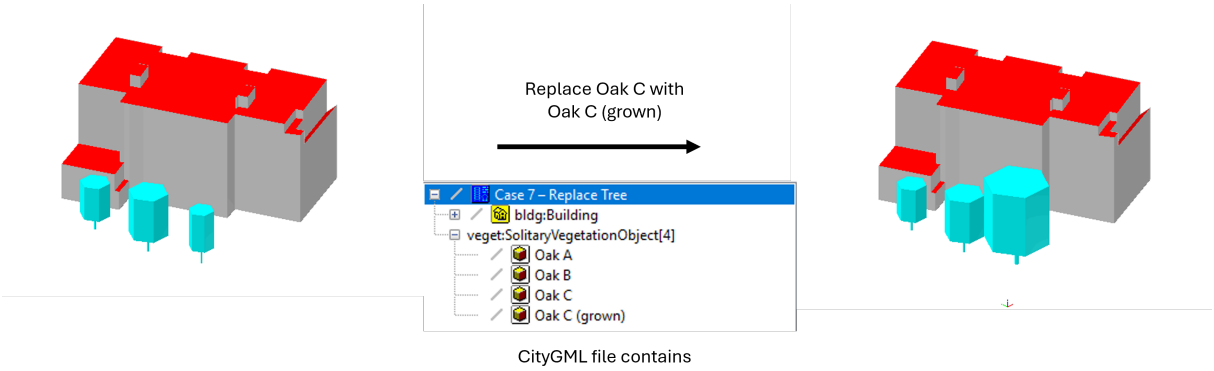


Figure 5.11: Operation 3 - Replacing a root feature (tree)

The XML encoding (Listing 3), shows the process of replacing the tree. This is very similar to the the two previous examples, except now the `FeatureTransactionType` is set to `replace`. The new tree, along with it's geometry and attributes in added in-line of the `VersionTransition`, as the `newFeature`. The `oldFeature` references the tree which is getting replaced. The `Version v2` contains the building and two standing oak tree's via the `clonePredecessor` flag set to `true`. The added tree is referenced through an xlink.

```
1   <?xml version='1.0' encoding='UTF-8' standalone='no'?>                                  xml
2   <core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
3     <gml:name>Case 3 – Replace Tree</gml:name>
4     <gml:boundedBy>
5     <!-- [envelope] -->
6     </gml:boundedBy>
7    <core:cityObjectMember>
8     <scn:Scenario gml:id="id_scenario_case3">                                  }
9       <gml:description>Oak C replaced with grown version reflecting current size</gml:description>
10      <gml:name>Case 3: Replace Tree</gml:name>
11      <scn:identifier>id_scenario_case3</scn:identifier>
12      <scn:validFrom>2036-01-01T00:00:00</scn:validFrom>
13      <scn:validTo>2060-12-31T23:59:59</scn:validTo>                                  } (1) Scenario
14      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">explorative</scn:type>
15      <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
16      <scn:version xlink:href="#id_version_case3_v2"/>
17      <scn:versionTransition xlink:href="#id_vt_case3"/>                          }
18    </scn:Scenario>
19  </core:cityObjectMember>
20  <gml:featureMember>
21    <scn:VersionTransition gml:id="id_vt_case3">                                  }
22      <gml:description>Oak C replaced with grown version</gml:description>
23      <scn:reason>It grew.</scn:reason>
24      <scn:clonePredecessor>true</scn:clonePredecessor>                           } (2) Version Transition
25      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">realized</scn:type>
26      <scn:from xlink:href="#id_version_case3_v1"/>
27      <scn:to xlink:href="#id_version_case3_v2"/>
28      <scn:featureTransaction>
29        <scn:FeatureTransaction gml:id="id_ft_case3_replace">
```

```
30          <gml:description>Replace Oak C with grown version</gml:description>
31          <scn:type codeSpace="http://.../codelists/FeatureTransactionType.xml">replace</scn:type>
32          <scn:oldFeature xlink:href="#VEG_Tree_Oak_003"/>
33          <scn:newFeature>
34            <veg:SolitaryVegetationObject gml:id="VEG_Tree_Oak_003_v2">
35              <!-- grown Oak C geometry and attributes omitted -->
36            </veg:SolitaryVegetationObject>
37          </scn:newFeature>
38        </scn:FeatureTransaction>
39      </scn:featureTransaction>
40    </scn:VersionTransition>
41  </gml:featureMember>
42  <gml:featureMember>
43    <scn:Version gml:id="id_version_case3_v1">
44      <gml:description>Building + 3 trees, original sizes</gml:description>
45      <scn:validFrom>2026-04-15T00:00:00</scn:validFrom>
46      <scn:tag>baseline</scn:tag>
47      <scn:versionMember xlink:href="#BAG_0599100000654922"/>
48      <scn:versionMember xlink:href="#VEG_Tree_Oak_001"/>
49      <scn:versionMember xlink:href="#VEG_Tree_Oak_002"/>
50      <scn:versionMember xlink:href="#VEG_Tree_Oak_003"/>
51    </scn:Version>
52  </gml:featureMember>
53  <gml:featureMember>
54    <scn:Version gml:id="id_version_case3_v2">
55      <gml:description>Building + 3 trees, Oak C updated to grown dimensions</gml:description>
56      <scn:validFrom>2036-01-01T00:00:00</scn:validFrom>
57      <scn:tag>scenario</scn:tag>
58      <!-- 654922, Oak_001 and Oak_002 carried forward from v1 by clonePredecessor -->
59      <scn:versionMember xlink:href="#VEG_Tree_Oak_003_v2"/>
60    </scn:Version>
61  </gml:featureMember>
62  <core:cityObjectMember>
63  <!--[1 Building (BAG_0599100000654922) and the 3 baseline Tree's (VEG_Tree_Oak_001, VEG_Tree_Oak_002, VEG_Tree_Oak_003)
    at root; the grown VEG_Tree_Oak_003_v2 sits inline in the transaction above] -->
64  </core:cityObjectMember>
65 </core:CityModel>
```

(3) Feature Transaction
(4) Version Transition
(5) Version v1 (baseline)
(6) Version v2 (Replaced Tree)

Listing 3: Replacing a Root Feature (tree)

5.2.1.4. Operation 4: Adding a child feature (solar panel)

For operation four the same applies as operation one, with the exception that we are now adding a child feature to an existing building. The operation can be visualised in Figure 5.12, where a solar panel is added to a building roof.

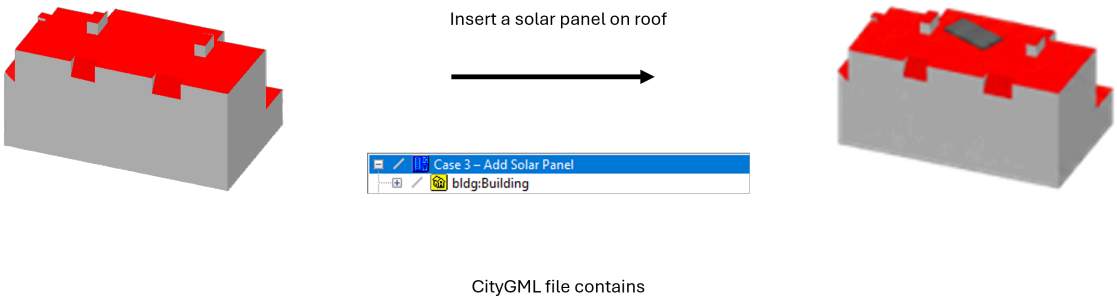


Figure 5.12: Operation 4 - Inserting a new child element (solar panel)

The XML example below shows the addition of a solar panel to the existing building. Similarly to the previous operations, it is important that the child feature sits in-line of the `versionTransition`. This can be seen in the `featureTransaction` (line 32), which has `type` `insert` with a `scn:newFeature` where the solar panel is represented in this case by a `bldg:BuildingInstallation`. The geometry and attributes are therefore also all within the transaction.

As usual, the `versionTransition` is followed by the `version` `v1` (baseline), `version` `v2` (version with added solar panel). As the solar panel is non root, only the building `BAG...5922` sits at root level in the file and can be seen at line 85 in Listing 4.

xml

```
1  <?xml version='1.0' encoding='UTF-8' standalone='no'?>
2  <core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
3    <gml:name>Case 4 – Add Solar Panel</gml:name>
4    <gml:boundedBy>
5      <!-- [envelope] -->
6    </gml:boundedBy>
7    <core:cityObjectMember>
8      <scn:Scenario gml:id="id_scenario_case4">
9        <gml:description>Rooftop solar panel added as child feature of building</
         gml:description>
10       <scn:identifier>id_scenario_case4</scn:identifier>
11       <scn:validFrom>2026-03-01T00:00:00</scn:validFrom>
12       <scn:validTo>2050-12-31T23:59:59</scn:validTo>
13       <scn:type codeSpace="http://.../codelists/ScenarioType.xml">explorative</scn:type>
14       <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
15       <scn:version xlink:href="#id_version_case4_v2"/>
16       <scn:versionTransition xlink:href="#id_vt_case4"/>
17     </scn:Scenario>
18   </core:cityObjectMember>
19   <gml:featureMember>
20     <scn:VersionTransition gml:id="id_vt_case4">
21       <gml:description>Solar panel installed</gml:description>
22       <scn:reason>Owner finally got the subsidy approved</scn:reason>
23       <scn:clonePredecessor>true</scn:clonePredecessor>
24       <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">realized</scn:type>
25       <scn:from xlink:href="#id_version_case4_v1"/>
26       <scn:to xlink:href="#id_version_case4_v2"/>
27       <scn:featureTransaction>
28         <scn:FeatureTransaction gml:id="id_ft_case4_insert">
29           <gml:description>Insert solar panel as child of building</gml:description>
30           <scn:type codeSpace="http://.../codelists/FeatureTransactionType.xml">insert</
             scn:type>
31           <scn:parentFeature xlink:href="#BAG_0599100000654922"/>
32           <scn:newFeature>
33             <bldg:BuildingInstallation gml:id="GEN_SolarPanel_001">
34               <gml:description>Rooftop PV installation</gml:description>
35               <gml:name>Solar panel – roof</gml:name>
36               <creationDate>2026-06-01</creationDate>
37               <bldg:lod2Geometry>
38                 <!-- Geometry of Solar Panel omitted -->
39               </bldg:lod2Geometry>
40               <gen:stringAttribute name="panelType">
41                 <gen:value>monocrystalline</gen:value>
42               </gen:stringAttribute>
43               <gen:intAttribute name="moduleCount">
44                 <gen:value>24</gen:value>
45               </gen:intAttribute>
46               <gen:doubleAttribute name="peakPower_kWp">
47                 <gen:value>8.4</gen:value>
48               </gen:doubleAttribute>
49               <gen:doubleAttribute name="tiltAngle_deg">
50                 <gen:value>30.0</gen:value>
51               </gen:doubleAttribute>
52               <gen:stringAttribute name="orientation">
53                 <gen:value>south</gen:value>
54               </gen:stringAttribute>
55             </bldg:BuildingInstallation>
56           </scn:newFeature>
57         </scn:FeatureTransaction>
58       </scn:featureTransaction>
59     </scn:VersionTransition>
60   </gml:featureMember>
61   <gml:featureMember>
62     <scn:Version gml:id="id_version_case4_v1">
```

(1) Scenario

(2) Version Transition

(3) Feature Transaction

(4) Version Transition

```
63        <gml:description>Building only, no solar panel</gml:description>
64        <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
65        <scn:tag>baseline</scn:tag>
66        <scn:versionMember xlink:href="#BAG_0599100000654922"/>
67      </scn:Version>
68    </gml:featureMember>
69    <gml:featureMember>
70      <scn:Version gml:id="id_version_case4_v2">
71        <gml:description>Building with solar panel installed</gml:description>
72        <scn:validFrom>2026-06-01T00:00:00</scn:validFrom>
73        <scn:tag>scenario</scn:tag>
74        <!-- BAG_0599100000654922 carried forward from v1 by clonePredecessor -->
75      </scn:Version>
76    </gml:featureMember>
77      <core:cityObjectMember>
78        <bldg:Building gml:id="BAG_0599100000654922">
79      </core:cityObjectMember>
80  </core:CityModel>
```

(5) Version v1 (baseline)

(6) Version v2 (Added solar panel)

Listing 4: Adding a child feature (solar panel)

5.2.1.5. Operation 5: Deleting a child feature (solar panel)

Deleting a child feature is achieved in a very similar way than the addition. The operation and members in the file can be seen below in Figure 5.13.

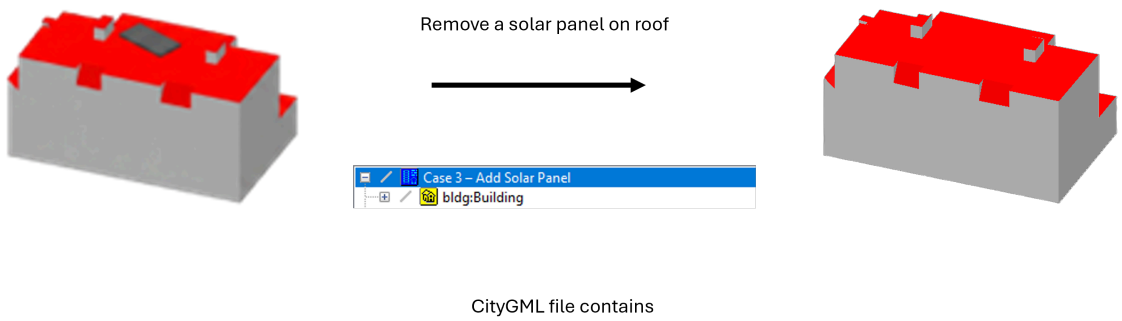


Figure 5.13: Operation 5 - Deleting a child feature (solar panel)

The XML encoding follows nearly an identical structure to operations involving the addition or removal of a root feature. It begins with a `Scenario` object which holds all the relevant metadata and links.

The `versionTransition` contains the `featureTransaction` which specifies the removal of the solar panel through the `scn:type` set to `delete`. The difference now between a root feature and a child feature is the use of the `scn:parentFeature` reference to the parent feature to which the panel belongs (line 31). The removed panel is referenced in the `scn:oldFeature`.

The rest of the encoding remains the same, both `version` are listed and members (in this case the `BAG...54922`) is present at root level in the file.

This entire encoding can be seen below in Listing 5.

```
1   <?xml version='1.0' encoding='UTF-8' standalone='no'?>                    xml
2   <core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
3     <gml:name>Case 5 – Remove Solar Panel</gml:name>
4     <gml:boundedBy>
5       <!-- [envelope] -->
6     </gml:boundedBy>
7     <core:cityObjectMember>
8       <scn:Scenario gml:id="id_scenario_case5">
9         <gml:description>Rooftop solar panel removed after end of service life</
          gml:description>
10        <scn:identifier>id_scenario_case5</scn:identifier>
11        <scn:validFrom>2051-01-01T00:00:00</scn:validFrom>
12        <scn:validTo>2055-12-31T23:59:59</scn:validTo>
```

```
13        <scn:type codeSpace="http://.../codelists/ScenarioType.xml">explorative</scn:type>
14        <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
15        <scn:version xlink:href="#id_version_case5_v2"/>
16        <scn:versionTransition xlink:href="#id_vt_case5"/>
17      </scn:Scenario>
18    </core:cityObjectMember>
19    <gml:featureMember>
20      <scn:VersionTransition gml:id="id_vt_case5">
21        <gml:description>Solar panel removed</gml:description>
22        <scn:reason>25 years of service, honourably discharged</scn:reason>
23        <scn:clonePredecessor>true</scn:clonePredecessor>
24        <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</
          scn:type>
25        <scn:from xlink:href="#id_version_case5_v1"/>
26        <scn:to xlink:href="#id_version_case5_v2"/>
27        <scn:featureTransaction>
28          <scn:FeatureTransaction gml:id="id_ft_case5_delete">
29            <gml:description>Delete solar panel from building</gml:description>
30            <scn:type codeSpace="http://.../codelists/FeatureTransactionType.xml">delete</
              scn:type>
31            <scn:parentFeature xlink:href="#BAG_0599100000654922"/>
32            <scn:oldFeature xlink:href="#GEN_SolarPanel_001"/>
33          </scn:FeatureTransaction>
34        </scn:featureTransaction>
35      </scn:VersionTransition>
36    </gml:featureMember>
37    <gml:featureMember>
38      <scn:Version gml:id="id_version_case5_v1">
39        <gml:description>Building with solar panel installed</gml:description>
40        <scn:validFrom>2026-06-01T00:00:00</scn:validFrom>
41        <scn:tag>baseline</scn:tag>
42        <scn:versionMember xlink:href="#BAG_0599100000654922"/>
43      </scn:Version>
44    </gml:featureMember>
45    <gml:featureMember>
46      <scn:Version gml:id="id_version_case5_v2">
47        <gml:description>Building without solar panel</gml:description>
48        <scn:validFrom>2051-01-01T00:00:00</scn:validFrom>
49        <scn:tag>scenario</scn:tag>
50        <!-- no versionMember: the building is cloned from v1, the panel removed by the delete
          transaction -->
51      </scn:Version>
52    </gml:featureMember>
53    <core:cityObjectMember>
54      <bldg:Building gml:id="BAG_0599100000654922">
55    </core:cityObjectMember>
56  </core:CityModel>
```

(1) Scenario
(2) Version Transition
(3) Feature Transaction
(4) Version Transition
(5) Version v1 (baseline)
(6) Version v2 (Removed solar panel)

Listing 5: Removing a child feature (solar panel)

5.2.1.6. Operation 6: Replacing a child feature (roof)

Operation six applies the replace pattern to a child feature. Where operation three swapped a root feature, here the `FeatureTransaction` targets a `bldg:RoofSurfac` e child of a building. The structure follows the same shape as the feature replacements above: type is set to `replace`, and both `scn:oldFeature` and `scn:newFeature` are referenced alongside `scn:parentFeature` linking back to the parent building. The geometry and attributes of the replacement surface sit inline within the transaction.

Encoding this case exposed a practical constraint specific to the 3DBAG dataset: a LOD2 roof is not one feature but a set of separate bldg:RoofSurface elements, one per detected roof plane. A single replace transaction handles one surface at a time, so replacing a full roof requires N deletes and one insert. The note below explains this in full. The encoding in Figure 5.14 and Listing 6 demonstrates the single-surface replace pattern.

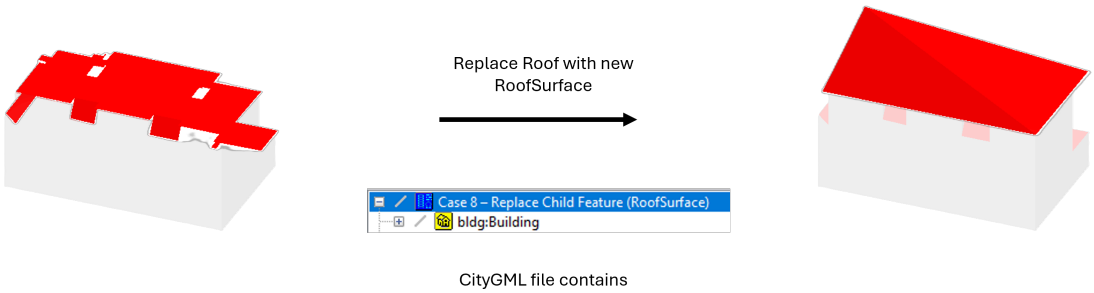


Figure 5.14: Operation 6 - Replacing a child feature (roof)

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <gml:name>Case 6 – Replace a roof</gml:name>
  <gml:boundedBy>
    <!-- [envelope] -->
  </gml:boundedBy>
  <core:cityObjectMember>
    <scn:Scenario gml:id="id_scenario_case6">
      <gml:description>Original roof surface replaced by a pyramid roof</gml:description>
      <scn:identifier>id_scenario_case6</scn:identifier>
      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
      <scn:validTo>2030-12-31T23:59:59</scn:validTo>
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">explorative</scn:type>
      <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
      <scn:version xlink:href="#id_version_case6_v2"/>
      <scn:versionTransition xlink:href="#id_vt_case6"/>
    </scn:Scenario>
  </core:cityObjectMember>
  <gml:featureMember>
    <scn:VersionTransition gml:id="id_vt_case6">
      <gml:description>Original roof replaced with pyramid roof</gml:description>
      <scn:reason>Roof renovation – surfaces unified into a pyramid design</scn:reason>
      <scn:clonePredecessor>true</scn:clonePredecessor>
      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</scn:type>
      <scn:from xlink:href="#id_version_case6_v1"/>
      <scn:to xlink:href="#id_version_case6_v2"/>
      <scn:featureTransaction>
        <scn:FeatureTransaction gml:id="id_ft_case6_replace_roof">
          <gml:description>Replace existing RoofSurface with pyramid roof</gml:description>
          <scn:type codeSpace="http://.../codelists/FeatureTransactionType.xml">replace</scn:type>
          <scn:parentFeature xlink:href="#BAG_0599100000654922"/>
          <scn:oldFeature xlink:href="#UUID_8d019743-972d-4e5e-8521-cd73dd6a79a1"/>
          <scn:newFeature>
            <bldg:RoofSurface gml:id="BLDG_FunkyRoof_001">
              <!-- pyramid roof geometry (multi-surface) omitted -->
            </bldg:RoofSurface>
          </scn:newFeature>
        </scn:FeatureTransaction>
      </scn:featureTransaction>
    </scn:VersionTransition>
  </gml:featureMember>
  <gml:featureMember>
    <scn:Version gml:id="id_version_case6_v1">
      <gml:description>Building with original multi-patch roof</gml:description>
      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
      <scn:tag>baseline</scn:tag>
      <scn:versionMember xlink:href="#BAG_0599100000654922"/>
      <scn:versionMember xlink:href="#UUID_8d019743-972d-4e5e-8521-cd73dd6a79a1"/>
    </scn:Version>
  </gml:featureMember>
  <gml:featureMember>
    <scn:Version gml:id="id_version_case6_v2">
      <gml:description>Building with new pyramid roof</gml:description>
```

(1) Scenario
(2) Version Transition
(3) Feature Transaction
(4) Version Transition
(5) Version v1 (baseline)

```
54        <scn:validFrom>2027-06-01T00:00:00</scn:validFrom>
55        <scn:tag>scenario</scn:tag>
56        <!-- 654922 carried forward from v1 by clonePredecessor; old roof removed by the replace
          transaction -->
57      </scn:Version>
58    </gml:featureMember>
59    <core:cityObjectMember>
60      <bldg:Building gml:id="BAG_0599100000654922">
61    </core:cityObjectMember>
62  </core:CityModel>
```

(6) Version v2 (Replaced roof)

Listing 6: Replacing a child feature (roof)

> **Note**
>
> This operation exposed one finding on how the 3DBAG constructs it's LOD2 geometries. The roof is not one feature but a set of separate `bldg:RoofSurface` elements, one per detected roof plane. A single replace transaction handles one surface at a time, so replacing a full roof requires N deletes and one insert. This would be redundant, and replacing the entire building would make more sense. This is brought up again in the discussion chapter.

5.2.1.7. Operation 7: Inserting a property attribute (U-value of a wall)

Operation seven introduces a different transaction type: the `AttributeTransaction`. Where all previous operations used a `FeatureTransaction` to add, remove, or replace whole features, attribute-level changes are handled through `scn:attributeTransaction` inside the `VersionTransition`.

The `AttributeTransaction` requires three fields beyond the type: `scn:parentFeature` pointing to the target building, `scn:attributeType`, and `scn:attributeName` naming the attribute to insert. The `scn:attributeType` is set to `cityGMLGenericAttribute`. A wall U-value is not a property defined by CityGML core or the Energy ADE, so it is carried as a generic attribute rather than a simple one. A simple attribute would only apply to a property already in the official schema, such as `function` or `measuredHeight`. For an insert, a `scn:value` is also required, holding the new value inside an `AttributeValue`. Because a U-value carries physical units, the value is encoded as `scn:measure` with a uom attribute. When an application applies this transaction, the attribute materialises on the building as a CityGML generic attribute. The kind of generic follows the value type. A `scn:measure` value becomes a `gen:measureAttribute`, so the U-value lands as a `gen:measureAttribute`. A unitless value such as the window-to-wall ratio in Operation 9 uses `scn:real` and lands as a `gen:doubleAttribute`.

As this operation does not lead to any visual change to the building no figure is shown. The full XML encoding is shown below in Listing 7.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <gml:name>Case 7 – Insert Wall U-value Attribute</gml:name>
  <gml:boundedBy>
    <!-- [envelope] -->
  </gml:boundedBy>
  <core:cityObjectMember>
    <scn:Scenario gml:id="id_scenario_case7">
      <gml:description>Wall U-value attribute added after post-renovation energy audit</
      gml:description>
      <scn:identifier>id_scenario_case7</scn:identifier>
      <scn:validFrom>2027-01-01T00:00:00</scn:validFrom>
      <scn:validTo>2030-12-31T23:59:59</scn:validTo>
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">explorative</scn:type>
      <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
      <scn:version xlink:href="#id_version_case7_v2"/>
      <scn:versionTransition xlink:href="#id_vt_case7"/>
    </scn:Scenario>
  </core:cityObjectMember>
  <gml:featureMember>
    <scn:VersionTransition gml:id="id_vt_case7">
```

(1) Scenario

```
21      <gml:description>Post-renovation wall U-value attribute inserted</gml:description>
22      <scn:reason>New in-situ measurement after facade insulation retrofit (ref:
        REN-2026-0311-followup)</scn:reason>
23      <scn:clonePredecessor>true</scn:clonePredecessor>
24      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">realized</
        scn:type>
25      <scn:from xlink:href="#id_version_case7_v1"/>
26      <scn:to xlink:href="#id_version_case7_v2"/>
27      <scn:attributeTransaction>
28        <scn:AttributeTransaction gml:id="id_at_case7_insert">
29          <gml:description>Insert post-renovation wallUValue_W_m2K from in-situ energy
            audit</gml:description>
30          <scn:type codeSpace="http://.../codelists/
            AttributeTransactionType.xml">insert</scn:type>
31          <scn:parentFeature xlink:href="#BAG_0599100000654922"/>
32          <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
33          <scn:attributeName>wallUValue_W_m2K</scn:attributeName>
34          <scn:value>
35            <scn:AttributeValue>
36              <scn:measure uom="W/(m2*K)">0.25</scn:measure>
37            </scn:AttributeValue>
38          </scn:value>
39        </scn:AttributeTransaction>
40      </scn:attributeTransaction>
41    </scn:VersionTransition>
42  </gml:featureMember>
43  <gml:featureMember>
44    <scn:Version gml:id="id_version_case7_v1">
45      <gml:description>Building without wall U-value attribute (pending new
        measurement)</gml:description>
46      <scn:validFrom>2026-06-01T00:00:00</scn:validFrom>
47      <scn:tag>baseline</scn:tag>
48      <scn:versionMember xlink:href="#BAG_0599100000654922"/>
49    </scn:Version>
50  </gml:featureMember>
51  <gml:featureMember>
52    <scn:Version gml:id="id_version_case7_v2">
53      <gml:description>Building with newly measured post-renovation wall U-value
        inserted</gml:description>
54      <scn:validFrom>2027-01-01T00:00:00</scn:validFrom>
55      <scn:tag>scenario</scn:tag>
56      <!-- no versionMember: the building is cloned from v1, the inserted attribute lives
        in the transaction -->
57    </scn:Version>
58  </gml:featureMember>
59  <core:cityObjectMember>
60    <bldg:Building gml:id="BAG_0599100000654922">
61  </core:cityObjectMember>
62 </core:CityModel>
```

(2) Version Transition

(3) Attribute Transaction

(4) Version Transition

(5) Version v1 (baseline)

(6) Version v2 (Inserted wall U-value)

Listing 7: Inserting a property attribute (U-value of a wall)

5.2.1.8. Operation 8: Deleting a property attribute (U-value of a wall)

Operation eight mirrors operation seven, with the difference that instead of inserting an attribute it removes one. The example use case here is that after a facade replacement the previously measured U-value is no longer valid and must be removed from the building record before a new measurement is taken.

The `versionTransition` contains an `attributeTransaction` with its type set to `delete`. Only the `scn:parentFeature` reference and the `scn:attributeName` are required. No `scn:value` is needed, as the attribute is being removed rather than set. The `clonePredecessor` flag is set to true, meaning all other attributes on the building carry over unchanged into v2. The full encoding is shown in Listing 8.

As this operation does not lead to any visual change to the building no figure is shown.

```xml
1  <?xml version='1.0' encoding='UTF-8' standalone='no'?>
2  <core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
3    <gml:name>Case 8 – Delete Wall U-value</gml:name>
```

```
4   <gml:boundedBy>
5     <!-- [envelope] -->
6   </gml:boundedBy>
7   <core:cityObjectMember>
8     <scn:Scenario gml:id="id_scenario_case8">
9       <gml:description>U-value attribute removed after facade replacement</
        gml:description>
10      <scn:identifier>id_scenario_case8</scn:identifier>
11      <scn:validFrom>2026-06-01T00:00:00</scn:validFrom>
12      <scn:validTo>2030-12-31T23:59:59</scn:validTo>
13      <scn:type codeSpace="http://.../1.0/codelists/ScenarioType.xml">explorative</
        scn:type>
14      <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
15      <scn:version xlink:href="#id_version_case8_v2"/>
16      <scn:versionTransition xlink:href="#id_vt_case8"/>
17    </scn:Scenario>
18  </core:cityObjectMember>
19  <gml:featureMember>
20    <scn:VersionTransition gml:id="id_vt_case8">
21      <gml:description>Stale wall U-value attribute deleted</gml:description>
22      <scn:reason>Facade replaced – pre-renovation U-value no longer valid, pending new
        measurement</scn:reason>
23      <scn:clonePredecessor>true</scn:clonePredecessor>
24      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</
        scn:type>
25      <scn:from xlink:href="#id_version_case8_v1"/>
26      <scn:to xlink:href="#id_version_case8_v2"/>
27      <scn:attributeTransaction>
28        <scn:AttributeTransaction gml:id="id_at_case8_delete">
29          <gml:description>Delete stale wallUValue_W_m2K generic attribute after facade
            replacement</gml:description>
30          <scn:type codeSpace="http://.../codelists/AttributeTransactionType.xml">delete</
            scn:type>
31          <scn:parentFeature xlink:href="#BAG_0599100000654922"/>
32          <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
33          <scn:attributeName>wallUValue_W_m2K</scn:attributeName>
34        </scn:AttributeTransaction>
35      </scn:attributeTransaction>
36    </scn:VersionTransition>
37  </gml:featureMember>
38  <gml:featureMember>
39    <scn:Version gml:id="id_version_case8_v1">
40      <gml:description>Building with pre-renovation wallUValue_W_m2K attribute</
        gml:description>
41      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
42      <scn:tag>baseline</scn:tag>
43      <scn:versionMember xlink:href="#BAG_0599100000654922"/>
44    </scn:Version>
45  </gml:featureMember>
46  <gml:featureMember>
47    <scn:Version gml:id="id_version_case8_v2">
48      <gml:description>Building with stale wall U-value attribute deleted after facade
        replacement</gml:description>
49      <scn:validFrom>2026-06-01T00:00:00</scn:validFrom>
50      <scn:tag>scenario</scn:tag>
51      <!-- no versionMember: the building is cloned from v1, the deleted attribute
        instruction lives in the transaction -->
52    </scn:Version>
53  </gml:featureMember>
54  <core:cityObjectMember>
55    <bldg:Building gml:id="BAG_0599100000654922">
56  </core:cityObjectMember>
57 </core:CityModel>
```

(1) Scenario
(2) Version Transition
(3) Attribute Transaction
(4) Version Transition
(5) Version v1 (baseline)
(6) Version v2 (Deleted wall U-value)

Listing 8: Deleting a property attribute (U-value of a wall)

5.2.1.9. Operation 9: Replacing a property attribute (WWR of a wall)

Operation nine updates an existing attribute to a new value. In this example the window-to-wall ratio of a south facade is revised upward after a theoretical daylighting improvement. This is the attribute-level equivalent of a root feature replace.

The `attributeTransaction` uses type set to `replace` and includes both `scn:attributeName` and a `scn:value` with the new value. As the WWR is a ratio rather than a measured quantity, the value is encoded as `scn:real` rather than `scn:measure`. The old value does not need to be referenced explicitly, however as a result the `scn:attributeName` must be encoded with the exact name of the attribute it wishes to replace. Otherwise a application layer applying the replacement would not know which attribute it is pointed towards. The full encoding is shown in Listing 9.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <gml:name>Case 9 – Replace Wall WWR</gml:name>
  <gml:boundedBy>
    <!-- [envelope] -->
  </gml:boundedBy>
  <core:cityObjectMember>
    <scn:Scenario gml:id="id_scenario_case9">
      <gml:description>Window-to-wall ratio updated after further analysis</gml:description>
      <scn:identifier>id_scenario_case9</scn:identifier>
      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
      <scn:validTo>2030-12-31T23:59:59</scn:validTo>
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">predictive</scn:type>
      <scn:projectName>ScenarioADE Beta6 – Test Suite</scn:projectName>
      <scn:version xlink:href="#id_version_case9_v2"/>
      <scn:versionTransition xlink:href="#id_vt_case9"/>
    </scn:Scenario>
  </core:cityObjectMember>
  <gml:featureMember>
    <scn:VersionTransition gml:id="id_vt_case9">
      <gml:description>Window-to-wall ratio updated</gml:description>
      <scn:reason>South facade glazing enlarged as part of daylighting improvement</scn:reason>
      <scn:clonePredecessor>true</scn:clonePredecessor>
      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</scn:type>
      <scn:from xlink:href="#id_version_case9_v1"/>
      <scn:to xlink:href="#id_version_case9_v2"/>
      <scn:attributeTransaction>
        <scn:AttributeTransaction gml:id="id_at_case9_wwr">
          <gml:description>Window-to-wall ratio updated from 7% to 20% after update</
          gml:description>
          <scn:type codeSpace="http://.../codelists/AttributeTransactionType.xml">replace</scn:type>
          <scn:parentFeature xlink:href="#BAG_0599100000618982"/>
          <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
          <scn:attributeName>windowToWallRatio</scn:attributeName>
          <scn:value>
            <scn:AttributeValue>
              <scn:real>0.2</scn:real>
            </scn:AttributeValue>
          </scn:value>
        </scn:AttributeTransaction>
      </scn:attributeTransaction>
    </scn:VersionTransition>
  </gml:featureMember>
  <gml:featureMember>
    <scn:Version gml:id="id_version_case9_v1">
      <gml:description>1 building, original window-to-wall ratio of 7%</gml:description>
      <scn:validFrom>2026-01-01T00:00:00</scn:validFrom>
      <scn:tag>baseline</scn:tag>
      <scn:versionMember xlink:href="#BAG_0599100000618982"/>
    </scn:Version>
  </gml:featureMember>
  <gml:featureMember>
    <scn:Version gml:id="id_version_case9_v2">
      <gml:description>1 building, window-to-wall ratio increased to 20% after upgrade</
      gml:description>
      <scn:validFrom>2026-07-01T00:00:00</scn:validFrom>
      <scn:tag>scenario</scn:tag>
      <!-- no versionMember: the building is cloned from v1, the changed attribute lives in the
      transaction -->
    </scn:Version>
```

(1) Scenario
(2) Version Transition
(3) Attribute Transaction
(4) Version Transition
(5) Version v1 (baseline)
(6) Version v2 (Replaced WWR)

```
  </gml:featureMember>
  <core:cityObjectMember>
    <bldg:Building gml:id="BAG_0599100000654922">
  </core:cityObjectMember>
</core:CityModel>
```

Listing 9: Replacing a property attribute (Window-to-Wall Ratio)

As this operation does not lead to any visual change to the building no figure is shown.

5.2.2. Storing the Scenario Results

The operations are shown to work and be succesful, however scenarios can also contain results which need storing. Therefore, when testing the Scenario ADE with reading and inserting the including result files the following cases arise which require some examples.

Before illustrating specific types of results and examples, it is important to provide an example of the general template required by the Scenario ADE to store simulation outputs. In line with the intentions, every result set must be contextualized with metadata detailing how it was generated.

As demonstrated in Listing 10 below, the general structure consists of:

- **Simulation Library**: Highlighted by the `Simulation Library` annotation, this serves as the root container. It encapsulates the `SimulationEngine` (specifying the software version, e.g., SimStadt) and the `Settings` (which hold the parameters used, such as calculation modes and weather files).
- **Version**: Highlighted by the `Version` annotation, a `scn:Version` element is instantiated to declare the state of the city model that the simulation was based upon.
- **Scenario**: Highlighted by the `Scenario` annotation, this element binds the context together. It is typed as `simulation-results`, points to the `Version`, and encapsulates the `ScenarioResults` container. The `ScenarioResults` element links back to the specific engine and settings used.

The actual output values (the transactions) are appended inside this `ScenarioResults` container. In the subsequent domain-specific examples, this verbose boilerplate will be abbreviated to focus purely on the transaction encodings.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <gml:name>Example - General Structure for Results</gml:name>
  <gml:boundedBy>
    <!-- Envelope -->
  </gml:boundedBy>
  <core:cityObjectMember>
    <scn:SimulationLibrary gml:id="lib_v3_test">
      <scn:simulationEngine>
        <scn:SimulationEngine gml:id="eng_v3_test">
          <gml:name>SimStadt</gml:name>
          <scn:version>3.0.0</scn:version>
        </scn:SimulationEngine>
      </scn:simulationEngine>
      <scn:settings>
        <scn:Settings gml:id="ss_v3_test">
          <scn:parameter>
            <scn:Parameter>
              <scn:name>calcMode</scn:name>
              <scn:value>
                <scn:AttributeValue>
                  <scn:string>HEATING_AND_COOLING</scn:string>
                </scn:AttributeValue>
              </scn:value>
            </scn:Parameter>
          </scn:parameter>
          <!-- Additional parameters omitted for example -->
        </scn:Settings>
      </scn:settings>
    </scn:SimulationLibrary>
  </core:cityObjectMember>
  <gml:featureMember>
```

(1) Simulation Library

```xml
      <scn:Version gml:id="ver_v3_test">
        <scn:versionMember xlink:href="#BAG_0599100000618982"/>
      </scn:Version>
    </gml:featureMember>
    <core:cityObjectMember>
      <scn:Scenario gml:id="scn_v3_test">
        <scn:type codeSpace="http://.../1.0/codelists/ScenarioType.xml">explorative</scn:type>
        <scn:projectName>Test Set Storing Results Example</scn:projectName>
        <scn:version xlink:href="#ver_v3_test"/>
        <scn:scenarioResults>
          <scn:ScenarioResults gml:id="sr_v3_test">
            <scn:simulationEngine xlink:href="#eng_v3_test"/>
            <scn:settings xlink:href="#ss_v3_test"/>
            <!-- Attribute and Feature Transactions go here -->
          </scn:ScenarioResults>
        </scn:scenarioResults>
      </scn:Scenario>
    </core:cityObjectMember>
  </core:CityModel>
```

(2) Version

(3) Scenario

Listing 10: General Boilerplate Structure for Storing Scenario Results

5.2.3. Domain Agnosticism and ADE dependencies

As one of the foundational intentions including keeping the ScenarioADE domain agnostic, there are different approaches that can be taken to correctly store results and scenarios using the Scenario ADE.

To display the ADE's versatility an example of how the same Scenario can be stored using the Energy ADE and using a generic method supported by the ADE is shown.

5.2.3.1. Storing Single Results

When storing single results, the Scenario ADE offers two valid ways to attach a value to a feature. The choice depends on whether the attribute is already defined by CityGML or a used ADE. A third option encodes the result natively through the Energy ADE.

The first, marked by the `Generic Attribute` annotation, uses a CityGML 2.0 generic attribute, with the `attributeType` set to `cityGMLGenericAttribute`. This is valid CityGML and breaks no XML rules. It is used here because `yearlyCoolingDemand` is not a property defined by CityGML or any ADE. A generic attribute is flexible and needs no schema change. Its drawbacks are the usual ones for generics. The semantics are weak and the value is limited to the fixed set of generic data types (string, integer, double, date, URI, and measure).

The second, marked by the `SimpleAttribute` annotation, sets a property that the schema already defines. Here it is `nrg3:bdgType` from the Energy ADE, written with its namespace so the defining schema can be checked. A simple attribute carries strong and interoperable semantics, but it can only be used for properties that CityGML or a used ADE already provides. It cannot be used for a custom result name such as `yearlyCoolingDemand`. For that reason the demand itself is stored either as the generic attribute above or as the Energy ADE feature below.

The third option, is the example of storing results based on the Energy ADE. Marked by the `Energy ADE` annotation, demonstrates how the same heating and cooling demand results can be encoded natively using the Energy ADE (`nrg3:Energy` elements) wrapped within a `scn:newFeatureTransaction`. This approach maintains strict domain coupling for energy results while still leveraging the Scenario ADE's versioning and transaction framework.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_v3_test">
      <scn:scenarioResults>
        <scn:ScenarioResults gml:id="sr_v3_test">
          <scn:simulationEngine xlink:href="#eng_v3_test"/>
          <scn:settings xlink:href="#ss_v3_test"/>
```

```
10
11            <scn:newAttributeTransaction>
12              <scn:AttributeTransaction gml:id="v3_BAG_0599100000618982_yearlyCoolingDemand">
13                <scn:type codeSpace="http://.../1.0/codelists/TransactionTypeValue.xml">insert</
                  scn:type>
14                <scn:parentFeature xlink:href="#BAG_0599100000618982"/>
15                <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
16                <scn:attributeName>yearlyCoolingDemand</scn:attributeName>
17                <scn:value>
18                  <scn:AttributeValue>
19                    <scn:measure uom="kWh/a">7800.0</scn:measure>
20                  </scn:AttributeValue>
21                </scn:value>
22              </scn:AttributeTransaction>
23            </scn:newAttributeTransaction>
24
25            <scn:newAttributeTransaction>
26              <scn:AttributeTransaction gml:id="v3s_BAG_0599100000618982_bdgType">
27                <scn:type codeSpace="http://.../1.0/codelists/TransactionTypeValue.xml">insert</
                  scn:type>
28                <scn:parentFeature xlink:href="#BAG_0599100000618982"/>
29                <scn:attributeType>simpleAttribute</scn:attributeType>
30                <scn:attributeName>nrg3:bdgType</scn:attributeName>
31                <scn:value>
32                  <scn:AttributeValue>
33                    <scn:string>residential</scn:string>
34                  </scn:AttributeValue>
35                </scn:value>
36              </scn:AttributeTransaction>
37            </scn:newAttributeTransaction>
38
39            <scn:newFeatureTransaction>
40              <scn:FeatureTransaction gml:id="ft_v3e_BAG_0599100000618982_yearlyCoolingDemand">
41                <scn:type codeSpace="http://.../codelists/TransactionTypeValue.xml">insert</scn:type>
42                <scn:parentFeature xlink:href="#BAG_0599100000618982"/>
43                <scn:newFeature>
44                  <nrg3:Energy gml:id="nrg_BAG_0599100000618982_yearlyCoolingDemand">
45                    <nrg3:operationType codeSpace="http://.../codelists/
                      ResourceOperationTypeValue.xml">demands</nrg3:operationType>
46                    <nrg3:amount uom="kWh/a">7800.0</nrg3:amount>
47                    <nrg3:year>2026</nrg3:year>
48                    <nrg3:isAmountNormalized>false</nrg3:isAmountNormalized>
49                    <nrg3:type codeSpace="http://.../codelists/EnergyTypeValue.xml">useful</
                      nrg3:type>
50                    <nrg3:endUse codeSpace="http://.../codelists/
                      EnergyEndUseValue.xml">spaceCooling</nrg3:endUse>
51                    <nrg3:energyCarrier codeSpace="http://.../codelists/
                      EnergyCarrierValue.xml">electricity</nrg3:energyCarrier>
52                  </nrg3:Energy>
53                </scn:newFeature>
54              </scn:FeatureTransaction>
55            </scn:newFeatureTransaction>
56
57          </scn:ScenarioResults>
58        </scn:scenarioResults>
59      </scn:Scenario>
60    </core:cityObjectMember>
61  <core:cityObjectMember>
62    <bldg:Building gml:id="BAG_0599100000618982">
63  </core:cityObjectMember>
64  </core:CityModel>
```

(1) Generic Attribute

(2) Simple Attribute

(3) Energy ADE

Listing 11: Storing Annual Heating and Cooling Demands as Domain Agnostic and Energy ADE

5.2.3.2. Storing Timeseries Results

A second common results type seen in the Scenarios is the Timeseries. Therefore this example is shown to address how to store such results as domain agnostic and through the EnergyADE.

In the case below, we illustrate two distinct approaches to capturing the monthly heating demand of a building over the simulated year. The first method, highlighted as `Generic Attribute`, stores the sequence of values as a space-separated `scn:measureList` in a generic attribute, with `attributeType` set to `cityGMLGenericAttribute`. This keeps the result domain agnostic, since `monthlyHeatingDemand` is not a property defined by CityGML or any ADE.

The second method, highlighted as `Energy ADE`, demonstrates a domain encoding where the timeseries is wrapped in an `nrg3:Energy` feature containing an `nrg3:RegularTimeSeries`. This structure strictly defines the temporal properties (e.g., start timestamp, time interval, and interpolation type).

> **Note**
>
> To keep the examples limited, not all Energy ADE attributes are shown. In a full implementation, these structures can be further expanded to capture the full breadth of domain specific data as required by the use case.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <cityObjectMember>
  <scn:Scenario gml:id="scn_v3ts_test">
    <scn:type codeSpace="http://.../codelists/ScenarioType.xml">exploratory</scn:type>
    <scn:projectName>3-Building Results Test — Feijenoord heating and cooling</scn:projectName>
    <scn:version xlink:href="#ver_v3ts_test"/>
    <scn:scenarioResults>
      <scn:ScenarioResults gml:id="sr_v3ts_test">
        <scn:simulationEngine xlink:href="#eng_v3ts_test"/>
        <scn:settings xlink:href="#ss_v3ts_test"/>

        <scn:newAttributeTransaction>
          <scn:AttributeTransaction gml:id="v3ts_BAG_0599100000618982_monthlyHeatingDemand">
            <scn:type codeSpace="http://.../codelists/TransactionTypeValue.xml">insert</scn:type>
            <scn:parentFeature xlink:href="#BAG_0599100000618982"/>
            <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
            <scn:attributeName>monthlyHeatingDemand</scn:attributeName>
            <scn:value>
              <scn:AttributeValue>
                <scn:measureList uom="kWh">7232.0 6328.0 5424.0 3164.0 1356.0 452.0 452.0 452.0 1356.0 3164.0 5424.0 10396.0</scn:measureList>
              </scn:AttributeValue>
            </scn:value>
          </scn:AttributeTransaction>
        </scn:newAttributeTransaction>

        <scn:newFeatureTransaction>
          <scn:FeatureTransaction gml:id="ft_v3rts_BAG_0599100000618982_monthlyHeatingDemand">
            <scn:type codeSpace="http://.../codelists/TransactionTypeValue.xml">insert</scn:type>
            <scn:parentFeature xlink:href="#BAG_0599100000618982"/>
            <scn:newFeature>
              <nrg3:Energy gml:id="nrg_ts_reg_BAG_0599100000618982_monthlyHeatingDemand">
                <nrg3:operationType codeSpace="http://.../codelists/ResourceOperationTypeValue.xml">demands</nrg3:operationType>
                <nrg3:amount uom="kWh/a">45200.0</nrg3:amount>
                <nrg3:year>2026</nrg3:year>
                <nrg3:isAmountNormalized>false</nrg3:isAmountNormalized>
                <nrg3:timeDependentAmount>
                  <nrg3:RegularTimeSeries gml:id="rts_BAG_0599100000618982_monthlyHeatingDemand">
                    <nrg3:interpolationType>averageInPrecedingInterval</nrg3:interpolationType>
                    <nrg3:startTimestamp>2026-01-01T00:00:00+01:00</nrg3:startTimestamp>
                    <nrg3:endTimestamp>2026-12-31T23:59:59+01:00</nrg3:endTimestamp>
                    <nrg3:timeInterval unit="other:month">1</nrg3:timeInterval>
                    <nrg3:valuesList uom="kWh">7232.0 6328.0 5424.0 3164.0 1356.0 452.0 452.0 452.0 1356.0 3164.0 5424.0 10396.0</nrg3:valuesList>
```

(1) Generic Attribute

```
45                    </nrg3:RegularTimeSeries>
46                  </nrg3:timeDependentAmount>
47                  <nrg3:type codeSpace="http://.../codelists/EnergyTypeValue.xml">useful</
                    nrg3:type>
48                  <nrg3:endUse codeSpace="http://.../codelists/
                    EnergyEndUseValue.xml">spaceHeating</nrg3:endUse>
49                  <nrg3:energyCarrier codeSpace="http://.../codelists/
                    EnergyCarrierValue.xml">heat</nrg3:energyCarrier>
50                </nrg3:Energy>
51              </scn:newFeature>
52            </scn:FeatureTransaction>
53          </scn:newFeatureTransaction>
54
55        </scn:ScenarioResults>
56      </scn:scenarioResults>
57    </scn:Scenario>
58  </cityObjectMember>
59  <cityObjectMember>
60    <bldg:Building gml:id="BAG_0599100000618982">
61  </cityObjectMember>
62 </CityModel>
```

(2) Energy ADE

Listing 12: Storing Timeseries Demands as Domain Agnostic and Energy ADE

5.2.4. Dealing with transaction duplication

A quick finding when running tests was the need to be able to support a 1 to many parent relationships. This was documented in the Scenario ADE Beta 6 micro-iterations. A encoded example of this adjustment is shown here, presented for the case of the *Refurbishment Scenario*.

For example, when encoding the refurbishment scenario a important pattern stood out. Due to the nature of physics-based energy simulations, and the limitation of lack of building data, the inherent reliance on archetypes becomes essential to enrich large urban areas. The simulations run in this thesis, 24 archetypes are used, leading to 24 combinations of thermal property data (U-values, etc). Therefore, when we have such repeating U-value enrichment into the feature transactions, redundancy is noticeable. A quick fix was through allowing 1 to many parent relationships on a feature and attribute transaction. Below, such a case is shown which becomes heavily utilised in the *Refurbishment Scenario (S3)* encoding at scale.

In this case these are shown as results, so added however they can equally be part of the `versionTransition` as part of the building attribute assumptions and changes needed to go from a previous `version` to the scenario `version`.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <cityObjectMember>
  <scn:Scenario gml:id="scn_v3ts_test">
    <scn:type codeSpace="http://.../codelists/ScenarioType.xml">exploratory</scn:type>
    <scn:projectName>Results Test – Feijenoord Insert U-values </scn:projectName>
    <scn:version xlink:href="#ver_v3ts_test"/>
    <scn:scenarioResults>
      <scn:ScenarioResults gml:id="sr_v3ts_test">
        <scn:simulationEngine xlink:href="#eng_v3ts_test"/>
        <scn:settings xlink:href="#ss_v3ts_test"/>

        <scn:newAttributeTransaction>
          <scn:AttributeTransaction
          gml:id="v06_at_wall_00_UUID_12032f1e-2892-44ee-96fb-30c0b80f7d68">
            <scn:type codeSpace="http://.../codelists/
            TransactionTypeValue.xml">insert</scn:type>
            <scn:parentFeature
            xlink:href="#UUID_12032f1e-2892-44ee-96fb-30c0b80f7d68"/>
```

```
18              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
19              <scn:attributeName>wallUValue</scn:attributeName>
20              <scn:value>                                                          } (1)
21                <scn:AttributeValue>                                               } (1)
22                  <scn:measure uom="W/(m2*K)">0.25</scn:measure>                   } (1)
23                </scn:AttributeValue>                                              } (1)
24              </scn:value>                                                         } (1)
25            </scn:AttributeTransaction>                                            } (1)
26          </scn:newAttributeTransaction>                                           } (1)
27          <scn:newAttributeTransaction>                                            } (1)
28            <scn:AttributeTransaction                                              } (1)
              gml:id="v06_at_wall_01_UUID_eb7b035e-75bf-4ff8-8537-                   } (1)
              e915befcabac">                                                         } (1)
29            <scn:type codeSpace="http://.../codelists/                             } (1)
              TransactionTypeValue.xml">insert</scn:type>                            } (1)
30            <scn:parentFeature xlink:href="#UUID_eb7b035e-75bf-4ff8-8537-          } (1)
              e915befcabac"/>                                                        } (1)
31            <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>         } (1)
32            <scn:attributeName>wallUValue</scn:attributeName>                      } (1)
33            <scn:value>                                                            } (1)
34              <scn:AttributeValue>                                                 } (1)
35                <scn:measure uom="W/(m2*K)">0.25</scn:measure>                     } (1)
36              </scn:AttributeValue>                                                } (1)
37            </scn:value>                                                           } (1)
38            </scn:AttributeTransaction>                                            } (1)
39          </scn:newAttributeTransaction>                                           } (1)
40
41          <scn:newAttributeTransaction>
42            <scn:AttributeTransaction
              gml:id="v06v2_at_walls_BAG_0599100000618982_wallUValue">
43            <scn:type codeSpace="http://.../codelists/                             } (2)
              TransactionTypeValue.xml">insert</scn:type>                            } (2)
44            <scn:parentFeature xlink:href="#UUID_XXX_30c0b80f7d68"/>                } (2)
45            <scn:parentFeature xlink:href="#UUID_XXX_e915befcabac"/>                } (2)
46            <scn:parentFeature xlink:href="#UUID_XXX_a09f6be89b07"/>                } (2)
47            <scn:parentFeature xlink:href="#UUID_XXX_31f3a3e35cf9"/>                } (2)
48            <scn:parentFeature xlink:href="#UUID_XXX_5e8b5e63eaec"/>                } (2)
49            <scn:parentFeature xlink:href="#UUID_XXX_85a069abcba7"/>                } (2)
50            <scn:parentFeature xlink:href="#UUID_XXX_f755e4afb009"/>                } (2)
51            <scn:parentFeature xlink:href="#UUID_XXX_c7589b073657"/>                } (2)
52            <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>         } (2)
53            <scn:attributeName>wallUValue</scn:attributeName>                      } (2)
54            <scn:value>                                                            } (2)
55              <scn:AttributeValue>                                                 } (2)
56                <scn:measure uom="W/(m2*K)">0.25</scn:measure>                     } (2)
57              </scn:AttributeValue>                                                } (2)
58            </scn:value>                                                           } (2)
59            </scn:AttributeTransaction>                                            } (2)
60          </scn:newAttributeTransaction>                                           } (2)
61
62        </scn:ScenarioResults>
63      </scn:scenarioResults>
64    </scn:Scenario>
65  </cityObjectMember>
66  <cityObjectMember>
67    <bldg:Building gml:id="BAG_0599100000618982">
68  </cityObjectMember>
69 </CityModel>
```

(1) Previous 0..1 parent relationship first 2/8 shown.

(2) 1 to many parent relationship

Listing 13: Storing U-values using Multiple Parents Vs Indiviual Feature Transactions

5.2.5. The 4 Case Study Scenarios

The following results prove the functionality and use of the Scenario ADE in encoding scenarios at neighborhood level. They utilise combinations of the shown operations above, as well as the storing of results.

For each scenario, the rows which address the scenario specific requirements defined at the start of the implementation chapter (Section 4.2.1.1) are highlighted. They additionally are tagged with the specific

requirement the encoded data refers to. With this traceability in mind, the are trimmed to display rows which do relate to the requirements, repetitive rows or trivial sections are omitted.

5.2.5.1. Future Climate Scenario

Applying the *Future Climate Scenario (S1)* to the neighborhood of Feijenoord included 1,677 buildings and their respective scenario results under a projected 2050 climate horizon. It is the simplest of the four scenarios in that no physical or attribute intervention is applied to the buildings themselves. Referring back to the requirements specified in Table 4, the encoding must prove that the ScenarioADE can fulfill the *context*, *configuration*, and *strategy* pillars. The seven data requirements that were specified in the table are therefore hihglighted in the color of the pillar they refer to, and given the tag of their requirement. (Listing 14)

The scenario narrative sits on the `Scenario` itself as a `gml:description` (S1-03), while `projectName` is kept only as a short grouping label. The contextual climate is carried both as a human-readable `weatherSource` string and as a `weatherFile` URI (`Rotterdam-mon-2050.tmy3`) so the input is both legible and resolvable (S1-01). Every `Parameter` additionally carries a `scope` (here `input`) and, where useful, a `description` , so a reader can tell a configuration input from a result without parsing names (C3).

The temporal horizon (S1-02) is attached to the `Scenario` , which is taken to own the window during which the scenario is assumed to hold. Here it is set to `validFrom` 2050 and `validTo` 2055; the five-year span is an arbitrary placeholder and could equally be left open with only a `validFrom` . The `Version` is treated as a static snapshot and carries only the `validFrom` of the state it represents (2050 for the future-climate version, 2025 for the baseline), while the `VersionTransition` carries no temporal window. The same horizon also travels as a `planningHorizonYear` parameter, so it is queryable as a value and not only as a timestamp (C2). The scenario is typed `exploratory` , and each `Version` is additionally tagged with its role (`baseline` or `scenario`) and its horizon year as lightweight metadata.

Two `attributeTransactions` are shown for the same building `(BAG_0599100000254436)` : the per-building results of the SimStadt simulation. As we assume no domain agnostic applicability, both use the `cityGMLGenericAttribute` type instead of the Energy ADE. The first records the yearly aggregate demand as a scalar `<scn:measure>` (3608 kWh/a heating), the second records the 12-month monthly profile as a `<scn:measureList>` , and the remaining 4,822 transactions in the file follow the same pattern across the other 1,205 buildings (each carrying yearly and monthly heating and cooling).

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_future_climate">
      <gml:description>Current Feijenoord stock evaluated under the KNMI 2050Hn future climate. No physical intervention is applied; only the climate context changes from the baseline.</gml:description>
      <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom> S1-02
      <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo> S1-02
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">predictive</scn:type>
      <scn:projectName>Feijenoord Future Climate</scn:projectName>
      <scn:version xlink:href="#v_future_climate"/>
      <scn:versionTransition xlink:href="#vt_future_climate"/>
      <scn:scenarioResults>
        <scn:ScenarioResults gml:id="sr_future_climate">
          <scn:simulationEngine xlink:href="#eng_simstadt"/>
          <scn:settings xlink:href="#ss_future_climate"/>
          <scn:newAttributeTransaction>
            <scn:AttributeTransaction gml:id="fc_BAG_0599100000254436_yearlyHeatingDemand">
              <scn:type>insert</scn:type>
              <scn:parentFeature xlink:href="#BAG_0599100000254436"/>
              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
              <scn:attributeName>yearlyHeatingDemand</scn:attributeName> S1-06
              <scn:value>
                <scn:AttributeValue>
```

```
25                  <scn:measure uom="kWh/a">3608.0</scn:measure>  S1-06
26                </scn:AttributeValue>
27              </scn:value>
28            </scn:AttributeTransaction>
29          </scn:newAttributeTransaction>
30          <scn:newAttributeTransaction>
31            <scn:AttributeTransaction gml:id="fc_ts_BAG_0599100000254436_monthlyHeatingDemand">
32              <scn:type>insert</scn:type>
33              <scn:parentFeature xlink:href="#BAG_0599100000254436"/>
34              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
35              <scn:attributeName>monthlyHeatingDemand</scn:attributeName>  S1-06
36              <scn:value>
37                <scn:AttributeValue>
38                  <scn:measureList uom="kWh">734 655 519 248 40 1 0 0 5 213 485 707</scn:measureList>
                    S1-06
39                </scn:AttributeValue>
40              </scn:value>
41            </scn:AttributeTransaction>
42          </scn:newAttributeTransaction>
43          <!-- 4822 more AttributeTransactions: 1206 buildings × (yearly H+C, monthly H+C) -->
44        </scn:ScenarioResults>
45      </scn:scenarioResults>
46    </scn:Scenario>
47  </core:cityObjectMember>
48  <core:cityObjectMember>
49    <scn:SimulationLibrary gml:id="lib_simstadt">
50      <scn:simulationEngine>
51        <scn:SimulationEngine gml:id="eng_simstadt">
52          <gml:name>SimStadt</gml:name>  S1-04
53          <scn:version>0.14.0-SNAPSHOT (develop, rev. ca81a73, 20251204)</scn:version>  S1-04
54        </scn:SimulationEngine>
55      </scn:simulationEngine>
56      <scn:settings>
57        <scn:Settings gml:id="ss_future_climate">
58          <scn:parameter>
59            <scn:Parameter>
60              <scn:scope>input</scn:scope>
61              <scn:description>Heating and cooling demand computed together (DIN 18599)</scn:description>
62              <scn:name>calcMode</scn:name>  S1-05
63              <scn:value>
64                <scn:AttributeValue>
65                  <scn:string>HEATING_AND_COOLING</scn:string>  S1-05
66                </scn:AttributeValue>
67              </scn:value>
68            </scn:Parameter>
69          </scn:parameter>
70          <scn:parameter>
71            <scn:Parameter>
72              <scn:scope>input</scn:scope>
73              <scn:description>KNMI 2050Hn future-climate projection for Rotterdam</scn:description>
74              <scn:name>weatherSource</scn:name>  S1-01
75              <scn:value>
76                <scn:AttributeValue>
77                  <scn:string>KNMI 2050Hn future climate</scn:string>  S1-01
78                </scn:AttributeValue>
79              </scn:value>
80            </scn:Parameter>
81          </scn:parameter>
82          <scn:parameter>
```

(1) AttributeTransaction (yearly)

(2) AttributeTransaction (monthly list)

```
83              <scn:Parameter>
84                <scn:scope>input</scn:scope>
85                <scn:description>Future-climate weather file (TMY3)</scn:description>
86                <scn:name>weatherFile</scn:name>
87                <scn:value>
88                  <scn:AttributeValue>
89                    <scn:uri>Rotterdam-mon-2050.tmy3</scn:uri>  S1-01
90                  </scn:AttributeValue>
91                </scn:value>
92              </scn:Parameter>
93            </scn:parameter>
94            <scn:parameter>
95              <scn:Parameter>
96                <scn:scope>input</scn:scope>
97                <scn:name>radiationModel</scn:name>  S1-05
98                <scn:value>
99                  <scn:AttributeValue>
100                   <scn:string>monthly_meteonorm_Perez</scn:string>  S1-05
101                 </scn:AttributeValue>
102               </scn:value>
103             </scn:Parameter>
104           </scn:parameter>
105           <scn:parameter>
106             <scn:Parameter>
107               <scn:scope>input</scn:scope>
108               <scn:name>physicsLibrary</scn:name>  S1-05
109               <scn:value>
110                 <scn:AttributeValue>
111                   <scn:string>Dutch Building Physics Library according to TABULA</scn:string>  S1-05
112                 </scn:AttributeValue>
113               </scn:value>
114             </scn:Parameter>
115           </scn:parameter>
116           <scn:parameter>
117             <scn:Parameter>
118               <scn:scope>input</scn:scope>
119               <scn:description>Scenario projection horizon (year)</scn:description>
120               <scn:name>planningHorizonYear</scn:name>  S1-02
121               <scn:value>
122                 <scn:AttributeValue>
123                   <scn:integer>2050</scn:integer>  S1-02
124                 </scn:AttributeValue>
125               </scn:value>
126             </scn:Parameter>
127           </scn:parameter>
128           <!-- location, crs, consumerBehaviour, simDate parameters omitted -->
129         </scn:Settings>
130       </scn:settings>
131     </scn:SimulationLibrary>
132   </core:cityObjectMember>
133   <gml:featureMember>
134     <scn:VersionTransition gml:id="vt_future_climate">  S1-07
135       <scn:reason>Future climate projection — same built environment, different climate forcing</
          scn:reason>
136       <scn:clonePredecessor>true</scn:clonePredecessor>  S1-07
137       <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">predicted</scn:type>  S1-07
138       <scn:from xlink:href="#v_baseline"/>
139       <scn:to xlink:href="#v_future_climate"/>
140       <!-- No feature/attribute transaction: null strategy (context-only change) -->  S1-07
141     </scn:VersionTransition>
142   </gml:featureMember>
143   <gml:featureMember>
144     <scn:Version gml:id="v_baseline">
145       <gml:description>Feijenoord baseline — 1677 buildings</gml:description>
```

(3) VersionTransition (null strategy)

```
146       <scn:validFrom>2025-01-01T00:00:00+01:00</scn:validFrom>  S1-02
147       <scn:tag>baseline</scn:tag>
148       <!-- versionMember xlink:href entries omitted -->
149     </scn:Version>
150   </cityObjectMember>
151   <cityObjectMember>
152     <scn:Version gml:id="v_future_climate">
153       <gml:description>Feijenoord under future climate — 1677 buildings, 2412 yearly + 2412 monthly
        attributes</gml:description>
154       <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom>  S1-02
155       <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo>  S1-02
156       <scn:tag>future-climate-2050</scn:tag>  S1-02
157       <scn:tag>scenario</scn:tag>
158       <!-- no versionMember: the buildings are cloned from v1, the changed attributes live in the
        transaction -->
159     </scn:Version>
160   </gml:featureMember>
161 </CityModel>
```

(4) Versions resolved by from/to xlinks

Listing 14: Abbreviated listing of *feijenoord_future_climate.gml* (on github). Highlights mark the lines where each requirement is located in the file: Context (S1-01, S1-02, S1-03), Config (S1-04, S1-05, S1-06), Strategy (S1-07).

5.2.5.2. Refurbishment Scenario

As introduced, the *Refurbishment Scenario (S2)* also builds upon the *Future Climate Scenario (S1)* horizon and re-runs the demand simulation after applying TABULA's *AdvancedRefurbishment* envelope upgrades to wall and roof U-values across the Feijenoord stock.

The schema must show that it can *modify* existing surface attributes (U-values) without duplicating their geometry, *label* the upgrade with a recognisable *standard identifier* so a reader knows which TABULA level was applied, and again store two families of result (yearly + monthly H/C demand). This time for the post-refurbishment version. As a reminder, Table 5 lists the seven requirements that the listing must satisfy, grouped by pillar.

Listing 15 shows an abbreviated listing of the scenario applied to all of Feijenoord, condensed from the full source file to one representative building, a few representative settings parameters, and one representative U-value cohort. Highlights are coloured by pillar and each carries a requirement tag referencing the table (Table 5).

Two distinct transaction patterns appear in this scenario. Inside `ScenarioResults` the same `cityGMLGenericAttribute` used in S1 stores the post-refurbishment yearly H/C demand (3312 kWh/ a heating shown) and the monthly profile (`<scn:measureList>`). Inside the `VersionTransition` itself, a separate `<scn:attributeTransaction>` block carries the actual *strategy*: one `AttributeTransaction` per U-value cohort, pointing at many surfaces via repeated `<scn:parentFeature>` xlinks, and asserting the new `uValueOpaqueElement` for all of them in one shot. The recognisable standard identifier (S2-03) is carried as a dedicated `refurbishmentStandard` parameter in the `Settings` (`TABULA Advanced`), rather than in the transition's free-text `reason`. This is the scaled version of dealing with transaction duplication (Section 5.2.4); the file applies 11,069 such surface-level U-value modifications, grouped into a far smaller number of cohorts (here only the first cohort of 19 surfaces at $U = 0.09\,W/m^2K$ is shown).

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_feijenoord_refurbishment">
      <gml:description>Feijenoord stock re-simulated under the inherited KNMI 2050Hn climate after applying TABULA
      AdvancedRefurbishment U-value upgrades to wall and roof surfaces.</gml:description>
      <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom>  S2-01
      <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo>  S2-01
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">exploratory</scn:type>
      <scn:projectName>Feijenoord Refurbishment</scn:projectName>
      <scn:version xlink:href="#v_refurbishment"/>
```

```
12        <scn:versionTransition xlink:href="#vt_refurbishment"/>
13        <scn:scenarioResults>
14          <scn:ScenarioResults gml:id="sr_refurbishment">
15            <scn:simulationEngine xlink:href="#eng_simstadt"/>
16            <scn:settings xlink:href="#ss_refurbishment"/>
17            <scn:newAttributeTransaction>
18              <scn:AttributeTransaction gml:id="rf_ID_7bcd1f6d_yearlyHeatingDemand">
19                <scn:type>insert</scn:type>
20                <scn:parentFeature xlink:href="#ID_7bcd1f6d"/>
21                <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
22                <scn:attributeName>yearlyHeatingDemand</scn:attributeName> S2-07
23                <scn:value>
24                  <scn:AttributeValue>
25                    <scn:measure uom="kWh/a">3312.0</scn:measure> S2-07
26                  </scn:AttributeValue>
27                </scn:value>
28              </scn:AttributeTransaction>
29            </scn:newAttributeTransaction>
30            <scn:newAttributeTransaction>
31              <scn:AttributeTransaction gml:id="rf_ts_ID_7bcd1f6d_monthlyHeatingDemand">
32                <scn:type>insert</scn:type>
33                <scn:parentFeature xlink:href="#ID_7bcd1f6d"/>
34                <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
35                <scn:attributeName>monthlyHeatingDemand</scn:attributeName> S2-07
36                <scn:value>
37                  <scn:AttributeValue>
38                    <scn:measureList uom="kWh">805 604 361 100 10 0 0 0 2 122 509 798</scn:measureList> S2-07
39                  </scn:AttributeValue>
40                </scn:value>
41              </scn:AttributeTransaction>
42            </scn:newAttributeTransaction>
43            <!-- 4822 more demand AttributeTransactions (1206 buildings × yearly+monthly × H+C) omitted -->
44          </scn:ScenarioResults>
45        </scn:scenarioResults>
46      </scn:Scenario>
47    </core:cityObjectMember>
48    <core:cityObjectMember>
49      <scn:SimulationLibrary gml:id="lib_simstadt">
50        <scn:simulationEngine>
51          <scn:SimulationEngine gml:id="eng_simstadt">
52            <gml:name>SimStadt</gml:name> S2-05
53            <scn:version>0.14.0-SNAPSHOT (develop, rev. d9a61c7, 20251028)</scn:version> S2-05
54          </scn:SimulationEngine>
55        </scn:simulationEngine>
56        <scn:settings>
57          <scn:Settings gml:id="ss_refurbishment">
58            <scn:parameter>
59              <scn:Parameter>
60                <scn:scope>input</scn:scope>
61                <scn:description>KNMI 2050Hn future-climate projection for Rotterdam</scn:description>
62                <scn:name>weatherSource</scn:name> S2-06
63                <scn:value>
64                  <scn:AttributeValue>
65                    <scn:string>KNMI 2050Hn future climate</scn:string> S2-01
66                  </scn:AttributeValue>
67                </scn:value>
68              </scn:Parameter>
69            </scn:parameter>
70            <scn:parameter>
71              <scn:Parameter>
72                <scn:scope>input</scn:scope>
73                <scn:description>Future-climate weather file (TMY3)</scn:description>
74                <scn:name>weatherFile</scn:name>
75                <scn:value>
76                  <scn:AttributeValue>
```

```
77                  <scn:uri>Rotterdam-mon-2050.tmy3</scn:uri>  S2-01
78                </scn:AttributeValue>
79              </scn:value>
80            </scn:Parameter>
81          </scn:parameter>
82          <scn:parameter>
83            <scn:Parameter>
84              <scn:scope>input</scn:scope>
85              <scn:name>radiationModel</scn:name>  S2-06
86              <scn:value>
87                <scn:AttributeValue>
88                  <scn:string>monthly_meteonorm_Perez</scn:string>  S2-06
89                </scn:AttributeValue>
90              </scn:value>
91            </scn:Parameter>
92          </scn:parameter>
93          <scn:parameter>
94            <scn:Parameter>
95              <scn:scope>input</scn:scope>
96              <scn:name>physicsLibrary</scn:name>  S2-06
97              <scn:value>
98                <scn:AttributeValue>
99                  <scn:string>Dutch Building Physics Library according to TABULA</scn:string>  S2-06
100               </scn:AttributeValue>
101             </scn:value>
102           </scn:Parameter>
103         </scn:parameter>
104         <scn:parameter>
105           <scn:Parameter>
106             <scn:scope>input</scn:scope>
107             <scn:description>TABULA refurbishment level applied to the envelope</scn:description>
108             <scn:name>refurbishmentStandard</scn:name>  S2-03
109             <scn:value>
110               <scn:AttributeValue>
111                 <scn:string>TABULA Advanced</scn:string>  S2-03
112               </scn:AttributeValue>
113             </scn:value>
114           </scn:Parameter>
115         </scn:parameter>
116         <scn:parameter>
117           <scn:Parameter>
118             <scn:scope>input</scn:scope>
119             <scn:description>Scenario projection horizon (year)</scn:description>
120             <scn:name>planningHorizonYear</scn:name>  S2-01
121             <scn:value>
122               <scn:AttributeValue>
123                 <scn:integer>2050</scn:integer>  S2-01
124               </scn:AttributeValue>
125             </scn:value>
126           </scn:Parameter>
127         </scn:parameter>
128         <!-- calcMode, location, crs, consumerBehaviour, simDate parameters omitted -->
129       </scn:Settings>
130     </scn:settings>
131   </scn:SimulationLibrary>
132 </core:cityObjectMember>
133 <gml:featureMember>
134   <scn:VersionTransition gml:id="vt_refurbishment">  S2-04
135     <scn:reason>Tabula AdvancedRefurbishment — wall/roof U-values updated to post-retrofit targets</scn:reason>
136     <scn:clonePredecessor>true</scn:clonePredecessor>  S2-04
137     <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</scn:type>  S2-04
138     <scn:from xlink:href="#v_baseline"/>
139     <scn:to xlink:href="#v_refurbishment"/>
```

```
140      <scn:attributeTransaction>
141        <scn:AttributeTransaction gml:id="at_surf_grp000_u0_0900"> S2-02
142          <scn:type>insert</scn:type>
143          <scn:parentFeature xlink:href="#UUID_a2edf053"/> S2-02
144          <scn:parentFeature xlink:href="#UUID_008f65ea"/> S2-02
145          <!-- 17 more parentFeature xlinks in this cohort (19 surfaces sharing post-refurb U = 0.09 W/
             (m²·K)) -->
146          <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
147          <scn:attributeName>uValueOpaqueElement</scn:attributeName> S2-02
148          <scn:value>
149            <scn:AttributeValue>
150              <scn:measure uom="W/(m2*K)">0.09</scn:measure> S2-02
151            </scn:AttributeValue>
152          </scn:value>
153        </scn:AttributeTransaction>
154      </scn:attributeTransaction>
155      <!-- More attributeTransaction cohorts for other U-value targets; 11069 surface updates in total -->
156    </scn:VersionTransition>
157  </gml:featureMember>
158  <gml:featureMember>
159    <scn:Version gml:id="v_baseline">
160      <gml:description>Feijenoord baseline — 1677 buildings</gml:description>
161      <scn:validFrom>2025-01-01T00:00:00+01:00</scn:validFrom> S2-01
162      <scn:tag>baseline</scn:tag>
163      <!-- 1677 versionMember xlink:href entries omitted -->
164    </scn:Version>
165  </gml:featureMember>
166  <gml:featureMember>
167    <scn:Version gml:id="v_refurbishment">
168      <gml:description>Feijenoord after Tabula AdvancedRefurbishment — 11069 surfaces, 1677 buildings</
         gml:description>
169      <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom> S2-01
170      <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo> S2-01
171      <scn:tag>refurbishment-2050</scn:tag> S2-01
172      <scn:tag>scenario</scn:tag>
173      <!-- no versionMember: the buildings are cloned from v1, the changed attributes live in the
         transaction -->
174    </scn:Version>
175  </gml:featureMember>
176 </CityModel>
```

(1) attributeTransaction (multi-surface U-value cohort)

(2) Versions resolved by from/to xlinks

Listing 15: Abbreviated listing of *feijenoord_refurbishment.gml*. Highlights mark the lines where each requirement is located in the file: Context (S2-01), Strategy (S2-02, S2-03, S2-04), Config (S2-05, S2-06, S2-07).

5.2.5.3. PV Adoption Scenario

As introduced, the *PV Adoption Scenario (S3)* builds upon the *Future Climate Scenario (S1)* and explores how much rooftop photovoltaic potential the Feijenoord building stock could yield under that future climate horizon.

Unlike S1, this scenario is declared *exploratory* (not predictive). The schema must show that it can reuse the same *simulation engine* under PV-specific calculation parameters and store two distinct families of result attached at the roof-surface level: a yearly per-surface yield (Double) and a monthly irradiance profile (List). Table 6 lists the six requirements that the listing must satisfy, grouped by pillar.

Listing 16 shows a condensed version from the full source file to one representative roof surface and a few representative settings parameters. Highlights are coloured by pillar and each carries a requirement tag reference the table (Table 6). The two `AttributeTransaction` blocks shown for surface `ID_94cf0938-…` follow the same `cityGMLGenericAttribute` pattern as S1: the first inserts a yearly scalar (4.82 MWh/a of PV potential), the second inserts a 12-element list (W/m² monthly irradiance). Both are attached directly to the roof surface that produced them. The `VersionTransition` is clone-only and carries no transaction, because the SimStadt PV run returns no new panel feature, only per-surface values. The remaining 2,192 transactions in the file repeat this pair across the other 1,096 roof surfaces out of all the

buildings returning a result in the simulation; together they constitute the `v_pv_adoption` version that the `VersionTransition` brings into being from `v_baseline`.

Note

- The SimStadt PV-potential run returns no panel feature or geometry, only per-surface potential values. The encoding therefore stores PV adoption purely as `AttributeTransaction` s on the existing roof `BuildingSurface` (`pvPotential` + `monthlyIrradianceOnRoof`) inside the `ScenarioResults` , and the `VersionTransition` does not carry any geometry. Feature and their geometries can however sit in-line of the `FeatureTransaction` so supporting the physical object with it's result is supported by the schema.

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_pv_adoption">
      <gml:description>Rooftop PV adoption explored for the Feijenoord stock under the inherited KNMI 2050Hn climate; per-surface PV potential yield is evaluated for each suitable roof surface.</gml:description>
      <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom> S3-01
      <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo> S3-01
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">exploratory</scn:type>
      <scn:projectName>Feijenoord PV Adoption</scn:projectName>
      <scn:version xlink:href="#v_pv_adoption"/>
      <scn:versionTransition xlink:href="#vt_pv_adoption"/>
      <scn:scenarioResults>
        <scn:ScenarioResults gml:id="sr_pv_adoption">
          <scn:simulationEngine xlink:href="#eng_simstadt"/>
          <scn:settings xlink:href="#ss_pv_adoption"/>
          <scn:newAttributeTransaction>
            <scn:AttributeTransaction gml:id="pv_ID_94cf0938_pvPotential">
              <scn:type>insert</scn:type>
              <scn:parentFeature xlink:href="#ID_94cf0938"/>
              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
              <scn:attributeName>pvPotential</scn:attributeName> S3-05
              <scn:value>
                <scn:AttributeValue>
                  <scn:measure uom="MWh/a">4.82</scn:measure> S3-05
                </scn:AttributeValue>
              </scn:value>
            </scn:AttributeTransaction>
          </scn:newAttributeTransaction>
          <scn:newAttributeTransaction>
            <scn:AttributeTransaction gml:id="pv_ts_ID_94cf0938_monthlyIrradianceOnRoof">
              <scn:type>insert</scn:type>
              <scn:parentFeature xlink:href="#ID_94cf0938"/>
              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
              <scn:attributeName>monthlyIrradianceOnRoof</scn:attributeName> S3-06
              <scn:value>
                <scn:AttributeValue>
                  <scn:measureList uom="W/m2">29 57 104 171 216 229 223 187 126 70 35 21</scn:measureList>
                  S3-06
                </scn:AttributeValue>
              </scn:value>
            </scn:AttributeTransaction>
          </scn:newAttributeTransaction>
          <!-- 2192 more AttributeTransactions: 1097 surfaces × (yearly pvPotential, monthly irradiance) -->
        </scn:ScenarioResults>
      </scn:scenarioResults>
    </scn:Scenario>
  </core:cityObjectMember>
  <core:cityObjectMember>
```

(1) AttributeTransaction (yearly scalar)

(2) AttributeTransaction (monthly list)

```
49      <scn:SimulationLibrary gml:id="lib_simstadt">
50        <scn:simulationEngine>
51          <scn:SimulationEngine gml:id="eng_simstadt">
52            <gml:name>SimStadt</gml:name> S3-03
53            <scn:version>0.14.0-SNAPSHOT (develop, rev. ca81a73, 20251204)</scn:version> S3-03
54          </scn:SimulationEngine>
55        </scn:simulationEngine>
56        <scn:settings>
57          <scn:Settings gml:id="ss_pv_adoption">
58            <scn:parameter>
59              <scn:Parameter>
60                <scn:scope>input</scn:scope>
61                <scn:description>KNMI 2050Hn future-climate projection for Rotterdam</scn:description>
62                <scn:name>weatherSource</scn:name> S3-04
63                <scn:value>
64                  <scn:AttributeValue>
65                    <scn:string>KNMI 2050Hn future climate</scn:string> S3-01
66                  </scn:AttributeValue>
67                </scn:value>
68              </scn:Parameter>
69            </scn:parameter>
70            <scn:parameter>
71              <scn:Parameter>
72                <scn:scope>input</scn:scope>
73                <scn:description>Future-climate weather file (TMY3)</scn:description>
74                <scn:name>weatherFile</scn:name>
75                <scn:value>
76                  <scn:AttributeValue>
77                    <scn:uri>Rotterdam-mon-2050.tmy3</scn:uri> S3-01
78                  </scn:AttributeValue>
79                </scn:value>
80              </scn:Parameter>
81            </scn:parameter>
82            <scn:parameter>
83              <scn:Parameter>
84                <scn:scope>input</scn:scope>
85                <scn:name>location</scn:name>
86                <scn:value>
87                  <scn:AttributeValue>
88                    <scn:string>Rotterdam</scn:string>
89                  </scn:AttributeValue>
90                </scn:value>
91              </scn:Parameter>
92            </scn:parameter>
93            <scn:parameter>
94              <scn:Parameter>
95                <scn:scope>input</scn:scope>
96                <scn:name>radiationModel</scn:name> S3-04
97                <scn:value>
98                  <scn:AttributeValue>
99                    <scn:string>monthly_meteonorm_Hay</scn:string> S3-04
100                 </scn:AttributeValue>
101               </scn:value>
102             </scn:Parameter>
103           </scn:parameter>
104           <scn:parameter>
105             <scn:Parameter>
106               <scn:scope>input</scn:scope>
107               <scn:description>Scenario projection horizon (year)</scn:description>
108               <scn:name>planningHorizonYear</scn:name> S3-01
109               <scn:value>
110                 <scn:AttributeValue>
111                   <scn:integer>2050</scn:integer> S3-01
112                 </scn:AttributeValue>
```

```
113              </scn:value>
114            </scn:Parameter>
115          </scn:parameter>
116          <!-- scenarioDescription, simDate parameters omitted -->
117        </scn:Settings>
118      </scn:settings>
119    </scn:SimulationLibrary>
120  </core:cityObjectMember>
121  <gml:featureMember>
122    <scn:VersionTransition gml:id="vt_pv_adoption">
123      <scn:reason>PV adoption scenario — rooftop PV potential evaluated per BuildingSurface</scn:reason>
124      <scn:clonePredecessor>true</scn:clonePredecessor>
125      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</scn:type>
126      <scn:from xlink:href="#v_baseline"/>
127      <scn:to xlink:href="#v_pv_adoption"/>
128      <!-- No feature/attribute transaction: PV is recorded as per-surface results, not a new feature -->
129    </scn:VersionTransition>
130  </gml:featureMember>
131  <gml:featureMember>
132    <scn:Version gml:id="v_baseline">
133      <gml:description>Feijenoord baseline — 1677 buildings</gml:description>
134      <scn:validFrom>2025-01-01T00:00:00+01:00</scn:validFrom> S3-01
135      <scn:tag>baseline</scn:tag>
136      <!-- 1677 versionMember xlink:href entries omitted -->
137    </scn:Version>
138  </gml:featureMember>
139  <gml:featureMember>
140    <scn:Version gml:id="v_pv_adoption">
141      <gml:description>Feijenoord with PV adoption — 1097 surfaces, 1097 pvPotential + 1097 monthly
         irradiance timeseries</gml:description>
142      <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom> S3-01
143      <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo> S3-01
144      <scn:tag>pv-adoption-2050</scn:tag> S3-01
145      <scn:tag>scenario</scn:tag>
146      <!-- no versionMember: the buildings are cloned from v1, the changed attributes live in the
         transaction. Since no added PV panel geometries none linked. -->
147    </scn:Version>
148  </gml:featureMember>
149 </CityModel>
```

(3) VersionTransition (clone-only)

(4) Versions resolved by from/to xlinks

Listing 16: Abbreviated listing of *feijenoord_pv_adoption.gml*. Highlights mark the lines where each requirement is located in the file: Context (S3-01), Config (S3-03, S3-04, S3-05, S3-06). The feature-addition requirement (S3-02) is discussed in the note above; the SimStadt PV run produces no new feature, so the listing carries no strategy transaction.

5.2.5.4. Urban Greening Scenario

As introduced, the *Urban Greening Scenario (S4)* again is based on the same *Future Climate Scenario (S1)* horizon and adds 1,599 street trees to the Feijenoord baseline to study their effect on rooftop PV yield and on building heating and cooling demand.

Of the four scenarios this is the most ambitious, combining every transaction pattern seen so far: post-greening H/C demand results (as in S1), per-roof PV yield after tree's (as in S3), and a `FeatureTransaction` that inserts trees into the existing city model (through xlink referencing).

The schema therefore shows that it can carry an explicit feature-addition strategy via `<scn:featureTransaction>` referencing the added `veg:SolitaryVegetationObject` CityGML objects, and host two different result sets (H/C demand and PV) under the same `ScenarioResults`. Table 7 lists the seven requirements that the listing must satisfy, grouped by pillar.

Listing 17 shows a condensed listing of *feijenoord_urban_greening.gml*. Highlights are coloured by pillar and each carries a requirement tag referencing the table (Table 7). Three distinct `AttributeTransaction` blocks appear under `ScenarioResults`: a yearly heating demand scalar (4936 kWh/a), a 12-element monthly heating list, and a yearly PV potential scalar (2.87 MWh/a). Together, they prove that one `ScenarioResults` can host both demand and generation results (the same pattern repeats for cooling and monthly irradiance, all omitted). Inside the `VersionTransition`, one `FeatureTransaction` per tree carries

the actual strategy through an in-line `<scn:newFeature>` `<veg:SolitaryVegetationObject>` skeleton. Each skeleton carries the height, trunk and crown diameter; the LOD2 geometry SimStadt uses for shading is kept in the source tree file and omitted here. The file is therefore self-contained while staying compact (one xlink and one lightweight skeleton per tree).

```xml
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_feijenoord_urban_greening">
      <gml:description>Street trees added to the Feijenoord baseline under the inherited KNMI 2050Hn climate, to study their effect on rooftop PV yield and on building heating and cooling demand.</gml:description>
      <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom> S4-01
      <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo> S4-01
      <scn:type codeSpace="http://.../codelists/ScenarioType.xml">exploratory</scn:type>
      <scn:projectName>Feijenoord Urban Greening</scn:projectName>
      <scn:version xlink:href="#v_urban_greening"/>
      <scn:versionTransition xlink:href="#vt_urban_greening"/>
      <scn:scenarioResults>
        <scn:ScenarioResults gml:id="sr_urban_greening">
          <scn:simulationEngine xlink:href="#eng_simstadt"/>
          <scn:settings xlink:href="#ss_urban_greening"/>
          <scn:newAttributeTransaction>
            <scn:AttributeTransaction gml:id="ug_ID_7bcd1f6d_yearlyHeatingDemand">
              <scn:type>insert</scn:type>
              <scn:parentFeature xlink:href="#ID_7bcd1f6d"/>
              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
              <scn:attributeName>yearlyHeatingDemand</scn:attributeName> S4-07
              <scn:value>
                <scn:AttributeValue>
                  <scn:measure uom="kWh/a">4936.0</scn:measure> S4-07
                </scn:AttributeValue>
              </scn:value>
            </scn:AttributeTransaction>
          </scn:newAttributeTransaction>
          <scn:newAttributeTransaction>
            <scn:AttributeTransaction gml:id="ug_ts_ID_7bcd1f6d_monthlyHeatingDemand">
              <scn:type>insert</scn:type>
              <scn:parentFeature xlink:href="#ID_7bcd1f6d"/>
              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
              <scn:attributeName>monthlyHeatingDemand</scn:attributeName> S4-07
              <scn:value>
                <scn:AttributeValue>
                  <scn:measureList uom="kWh">1076 861 641 260 38 2 0 0 20 275 718 1044</scn:measureList>
                  S4-07
                </scn:AttributeValue>
              </scn:value>
            </scn:AttributeTransaction>
          </scn:newAttributeTransaction>
          <scn:newAttributeTransaction>
            <scn:AttributeTransaction gml:id="ug_UUID_81b18957_pvPotential">
              <scn:type>insert</scn:type>
              <scn:parentFeature xlink:href="#UUID_81b18957"/>
              <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
              <scn:attributeName>pvPotential</scn:attributeName> S4-06
              <scn:value>
                <scn:AttributeValue>
                  <scn:measure uom="MWh/a">2.87</scn:measure> S4-06
                </scn:AttributeValue>
              </scn:value>
            </scn:AttributeTransaction>
          </scn:newAttributeTransaction>
          <!-- 6873 more demand + PV result AttributeTransactions (yearly+monthly H/C + per-surface PV+irradiance) omitted -->
```

(1) AttributeTransaction (yearly H demand)

(2) AttributeTransaction (monthly H demand)

(3) AttributeTransaction (yearly PV yield)

```
57          </scn:ScenarioResults>
58        </scn:scenarioResults>
59      </scn:Scenario>
60    </core:cityObjectMember>
61    <core:cityObjectMember>
62      <scn:SimulationLibrary gml:id="lib_simstadt">
63        <scn:simulationEngine>
64          <scn:SimulationEngine gml:id="eng_simstadt">
65            <gml:name>SimStadt</gml:name> S4-04
66            <scn:version>0.14.0-SNAPSHOT (develop, rev. d5fb05c, 20250715)</scn:version> S4-04
67          </scn:SimulationEngine>
68        </scn:simulationEngine>
69        <scn:settings>
70          <scn:Settings gml:id="ss_urban_greening">
71            <scn:parameter>
72              <scn:Parameter>
73                <scn:scope>input</scn:scope>
74                <scn:description>KNMI 2050Hn future-climate projection for Rotterdam</scn:description>
75                <scn:name>weatherSource</scn:name> S4-05
76                <scn:value>
77                  <scn:AttributeValue>
78                    <scn:string>KNMI 2050Hn future climate</scn:string> S4-01
79                  </scn:AttributeValue>
80                </scn:value>
81              </scn:Parameter>
82            </scn:parameter>
83            <scn:parameter>
84              <scn:Parameter>
85                <scn:scope>input</scn:scope>
86                <scn:description>Future-climate weather file (TMY3)</scn:description>
87                <scn:name>weatherFile</scn:name>
88                <scn:value>
89                  <scn:AttributeValue>
90                    <scn:uri>Rotterdam-mon-2050.tmy3</scn:uri> S4-01
91                  </scn:AttributeValue>
92                </scn:value>
93              </scn:Parameter>
94            </scn:parameter>
95            <scn:parameter>
96              <scn:Parameter>
97                <scn:scope>input</scn:scope>
98                <scn:name>radiationModel</scn:name> S4-05
99                <scn:value>
100                 <scn:AttributeValue>
101                   <scn:string>monthly_meteonorm_Hay</scn:string> S4-05
102                 </scn:AttributeValue>
103               </scn:value>
104             </scn:Parameter>
105           </scn:parameter>
106           <scn:parameter>
107             <scn:Parameter>
108               <scn:scope>input</scn:scope>
109               <scn:name>physicsLibrary</scn:name> S4-05
110               <scn:value>
111                 <scn:AttributeValue>
112                   <scn:string>Dutch Building Physics Library according to TABULA</scn:string> S4-05
113                 </scn:AttributeValue>
114               </scn:value>
115             </scn:Parameter>
116           </scn:parameter>
117           <scn:parameter>
118             <scn:Parameter>
119               <scn:scope>input</scn:scope>
120               <scn:name>interventionType</scn:name> S4-02
```

```
121              <scn:value>
122                <scn:AttributeValue>
123                  <scn:string>streetTreesPlanting</scn:string> S4-02
124                </scn:AttributeValue>
125              </scn:value>
126            </scn:Parameter>
127          </scn:parameter>
128          <scn:parameter>
129            <scn:Parameter>
130              <scn:scope>input</scn:scope>
131              <scn:description>Scenario projection horizon (year)</scn:description>
132              <scn:name>planningHorizonYear</scn:name> S4-01
133              <scn:value>
134                <scn:AttributeValue>
135                  <scn:integer>2050</scn:integer> S4-01
136                </scn:AttributeValue>
137              </scn:value>
138            </scn:Parameter>
139          </scn:parameter>
140          <!-- calcMode, location, crs, consumerBehaviour, simDate parameters omitted -->
141        </scn:Settings>
142      </scn:settings>
143    </scn:SimulationLibrary>
144  </core:cityObjectMember>
145  <gml:featureMember>
146    <scn:VersionTransition gml:id="vt_urban_greening">
147      <scn:reason>Urban greening — street trees added per municipality plan</scn:reason>
148      <scn:clonePredecessor>true</scn:clonePredecessor>
149      <scn:type codeSpace="http://.../codelists/VersionTransitionType.xml">planned</scn:type>
150      <scn:from xlink:href="#v_baseline"/>
151      <scn:to xlink:href="#v_urban_greening"/>
152      <scn:featureTransaction> S4-02
153        <scn:FeatureTransaction gml:id="ft_ug_bm_68737_K"> S4-02
154          <scn:type>insert</scn:type> S4-02
155          <scn:newFeature> S4-02
156            <veg:SolitaryVegetationObject gml:id="bm_68737_K"> S4-02
157              <veg:height uom="m">8.0</veg:height> S4-02
158              <veg:trunkDiameter uom="cm">16.0</veg:trunkDiameter> S4-02
159              <veg:crownDiameter uom="m">5.0</veg:crownDiameter> S4-02
160              <!-- veg:lod2Geometry kept in the source tree file, omitted here--> S4-02
161            </veg:SolitaryVegetationObject> S4-02
162          </scn:newFeature>
163        </scn:FeatureTransaction>
164      </scn:featureTransaction> S4-01
165      <!-- 1598 more FeatureTransactions (one per tree) omitted -->
166    </scn:VersionTransition>
167  </gml:featureMember>
168  <gml:featureMember>
169    <scn:Version gml:id="v_baseline">
170      <gml:description>Feijenoord baseline — 1677 buildings</gml:description>
171      <scn:validFrom>2025-01-01T00:00:00+01:00</scn:validFrom>
172      <scn:tag>baseline</scn:tag> S4-01
173      <!-- 1677 versionMember xlink:href entries omitted --> S4-01
174    </scn:Version> S4-01
175  </gml:featureMember>
176  <gml:featureMember>
177    <scn:Version gml:id="v_urban_greening">
178      <gml:description>Feijenoord with urban greening — 1078 buildings + 1599 trees, 3438 yearly + 3438
       timeseries attributes</gml:description>
179      <scn:validFrom>2050-01-01T00:00:00+01:00</scn:validFrom>
180      <scn:validTo>2055-12-31T23:59:59+01:00</scn:validTo>
```

(4) FeatureTransaction (insert tree)

(5) Versions resolved by from/to xlinks

```
      <scn:tag>urban-greening-2050</scn:tag>
      <scn:tag>scenario</scn:tag>
      <!-- 1078 building inherited via clonepredecessor, 1599 tree versionMember xlink:href entries omitted -->
    </scn:Version>
  </gml:featureMember>
</CityModel>
```

Listing 17: Abbreviated listing of *feijenoord_urban_greening.gml*. Highlights mark the lines where each requirement is located in the file: Context (S4-01), Strategy (S4-02), Config (S4-04, S4-05, S4-06, S4-07).

5.2.5.5. The Scenario Multiverse in a CityGML file

This final encoding brings the baseline and all four scenarios together in one CityGML file. It shows that a single document can hold a whole set of related scenarios instead of one in isolation, which is what makes scenarios useful to compare (F2). Listing 18 shows the structure, with the four sections annotated in the same way as the scenario listings before it.

The file groups its members by type. First come the four `Scenario` objects, each still holding its own results. Then a single `SimulationLibrary` holds one shared `SimulationEngine` and four `Settings`, one per scenario, so the engine is defined once and reused across all of them (CF1). Next are the four `VersionTransition` objects that hold the modifications. Last are the five `Version` objects, the baseline plus one per scenario, followed by the buildings, building parts and trees that the versions reference.

The tags record the lineage. Every scenario `Version` carries the `scenario` tag next to its horizon tag, and only `v_baseline` carries the `baseline` tag. The transitions then form a small tree. The Future Climate transition is the only one whose `from` still points to the baseline-tagged version. The Refurbishment, PV Adoption and Urban Greening transitions all branch from `v_future_climate` instead, since each is evaluated under the same 2050 climate. So the baseline feeds the Future Climate, and the Future Climate in turn feeds the three interventions.

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>                                    xml
<core:CityModel xmlns="http://www.opengis.net/citygml/2.0" ....>
  <!-- Boilerplate omitted (see Listing 13) -->
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_future_climate">
    <!-- Future Climate Scenario-->
  </core:cityObjectMember>
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_pv_adoption">                                    } (1) The 4 Scenarios
    <!-- PV Adoption Scenario-->
  </core:cityObjectMember>
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_refurbishment">
    <!-- Refurbishment Scenario-->
  </core:cityObjectMember>
  <core:cityObjectMember>
    <scn:Scenario gml:id="scn_urban_greening">
    <!-- Urban Greening Scenario-->
  </core:cityObjectMember>
    <scn:SimulationLibrary gml:id="lib_simstadt">
      <scn:simulationEngine>
        <scn:SimulationEngine gml:id="eng_simstadt">
        <!-- SimStadt Engine (name and version) -->
      </scn:simulationEngine>
      <scn:settings>
        <scn:Settings gml:id="ss_future_climate">
        <!-- Future Climate Scenario Settings) -->
      </scn:settings>
      <scn:settings>
        <scn:Settings gml:id="ss_refurbishment">
        <!-- Refurbishment Scenario Settings) -->
      </scn:settings>
      <scn:settings>
        <scn:Settings gml:id="ss_pv_adoption">
        <!-- PV Adoption Scenario Settings) -->
      </scn:settings>
```

```
37          <scn:settings>
38            <scn:Settings gml:id="ss_urban_greening">
39            <!-- Urban Greening Scenario Settings) -->
40          </scn:settings>
41        </scn:SimulationLibrary>
42      </core:cityObjectMember>
43      <gml:featureMember>
44        <scn:VersionTransition gml:id="vt_future_climate">
45        <!-- Future Climate Scenario -->
46      </gml:featureMember>
47      <gml:featureMember>
48        <scn:VersionTransition gml:id="vt_refurbishment">
49        <!-- Refurbishment Scenario -->
50      </gml:featureMember>
51      <gml:featureMember>
52        <scn:VersionTransition gml:id="vt_pv_adoption">
53        <!-- PV Adoption Scenario -->
54      </gml:featureMember>
55      <gml:featureMember>
56        <scn:VersionTransition gml:id="vt_urban_greening">
57        <!-- Urban Greening Scenario -->
58      </gml:featureMember>
59      <gml:featureMember>
60        <scn:Version gml:id="v_baseline">
61        <!-- Baseline Version -->
62      </gml:featureMember>
63      <gml:featureMember>
64        <scn:Version gml:id="v_future_climate">
65        <!-- Future Climate Version -->
66      </gml:featureMember>
67      <gml:featureMember>
68        <scn:Version gml:id="v_refurbishment">
69        <!-- Refurbishment Version -->
70      </gml:featureMember>
71      <gml:featureMember>
72        <scn:Version gml:id="v_pv_adoption">
73        <!-- PV Adoption Version -->
74      </gml:featureMember>
75      <gml:featureMember>
76        <scn:Version gml:id="v_urban_greening">
77        <!-- Urban Greening Version -->
78      </gml:featureMember>
79      <core:cityObjectMember>
80        <bldg:Building gml:id="BAG_0599100100018100">
81        <!-- All buildings, tree's belonging to the different versions (omitted) -->
82        </bldg:Building>
83      </core:cityObjectMember>
84    </core:CityModel>
```

(2) SimulationLibrary (1 Engine & 4 Settings)

(3) The 4 VersionTransitions

(4) The 5 Versions (baseline + 4 scenarios)

Listing 18: Storing the case study scenarios in a multiverse

5.2.6. File Size Comparison of Multiverse vs Individual Scenario

A comparison of how much size is saved through using the Scenario ADE for the same scenarios and results encoded as individual scenarios, and all together is shown in Table 15.

Category	File	Lines	Bytes	Smaller vs. sum (%)
Individual	Future Climate (S1)	120,626	6,380,915	–
Individual	PV Adoption (S2)	86,391	4,310,510	–
Individual	Refurbishment (S3)	133,775	7,462,885	–
Individual	Urban Greening (S4)	169,699	9,178,187	–
Sum of individual	(4 files combined)	510,491	27,332,497	–
Multiverse	Scenario Multiverse	332,836	19,490,515	–
Savings (multiverse vs. sum)		177,655	7,841,982	28.7%

Table 15: File size comparison between storing the four case study scenarios as individual CityGML files versus a single multiverse file.

Each individual file carries the full Feijenoord base model of 1677 buildings together with one scenario. The multiverse carries the same 1677 buildings and adds all four scenarios on top. The building model is identical in both encodings, with the same set of `bldg:` elements present in a single individual file and in the multiverse, so the geometric and thematic detail is preserved exactly. The multiverse also keeps every scenario relationship. It holds the four scenarios, their four `VersionTransition` records and the five `Version` objects that capture the lineage from the shared baseline. The reason it is smaller than the sum of the individual files is that an individual file has to repeat the whole base model for its single scenario. When the four files are stored together the base model is written four times over. The multiverse writes the base model once and branches the four scenarios from it. What the multiverse removes is the duplicated baseline, not the detail and not the lineage.

5.2.7. SQL Results

The SQL encoding demonstrates that the Scenario ADE functions according to its foundational design principles Table 1. Through showing output tables returned by SQL queries, the results provide evidence on how to extract specific information while also highlighting the schema's underlying structure. The examples can also be interpreted as to illustrate how an application layer might interact with the database to retrieve necessary data and present it to the user through a Graphical User Interface (GUI).

Therefore, 3DCityDB instance is first enriched with four scenarios influencing a total of 10 buildings. This is used to show the results on a small sample that is easy to understand. The last result shows the database population scaled to the entire neighborhood of Feijenoord, matching the content of the *Scenario Multiverse* file, and showing that the database can indeed host multiple large scale scenarios in one environment.

For reproduceability, all of the SQL queries constructed can be found in the Appendix (Section D)

5.2.7.1. What is in the database?

Here, it is shown that all classes of the Scenario ADE are present in the database. For the filtered 10 buildings in Feijenoord the database contained the rows shown in Table 16. This representation also perfectly matches what the *Scenario Multiverse* file contained, where each scenario has it's own `VersionTransition`, `Settings` and `ScenarioResults`. Additionally there are five `Version` matching the baseline and the four scenarios.

#	objectclass_id	class_name	n_rows
1	20100	Version	5
2	20110	VersionTransition	4
3	20120	FeatureTransaction	40
4	20130	AttributeTransaction	309
5	20200	SimulationLibrary	1
6	20210	SimulationEngine	1
7	20220	Settings	4
8	20300	Scenario	4
9	20310	ScenarioResults	4

Table 16: Populated Database with all four Scenarios

5.2.7.2. Reusability of Scenario Settings and Simulation Engines

The first test shown addresses the question from principle CF1: *If I ran the same setup across multiple scenarios, would the configuration need to be defined every time?*.

The query resulting in the results table below tests whether the settings and the simulation engine are seperable reusable entities, avoiding entire duplication. This was hinted at in the population of the database where we can see that there is just one `SimulationEngine` present and four `Settings`. Below, Table 17 shows the same, each `Scenario` utilises the same `engine` and their own unique `settings`.

#	scenario	engine	settings
1	scn_future_climate	eng_simstadt	ss_future_climate
2	scn_pv_adoption	eng_simstadt	ss_pv_adoption
3	scn_refurbishment	eng_simstadt	ss_refurbishment
4	scn_urban_greening	eng_simstadt	ss_urban_greening

Table 17: Results of the shared engine test.

If the same `Settings` were also shared across multiple scenarios, then not only the simulation engine but also the settings would be counted once. The schema permits this: `Settings` is modelled as a feature (it extends `AbstractFeatureWithLifeSpan` and is mapped to its own row in `scn_simulation` with its own `gml_id`), and `SettingsPropertyType` carries `gml:AssociationAttributeGroup`, so multiple `Scenario` instances could in principle reference the same `Settings` by `xlink:href`. For the case study, however, all four scenarios were enriched through their own parameters, so each receives its own distinct `Settings` instance.

5.2.7.3. Results are Traceable

This SQL example addresses the question from principle CF2: *Are the results traceable back to the specific object and simulation run that produced them?*

For every result attribute the query returns the chain back to what produced it. It reaches the building that owns the value, the `AttributeTransaction` that holds it, the `ScenarioResults` it belongs to, the `Scenario`, and the `SimulationEngine` and `Settings` that defined the run. These are all linked through foreign keys, so a single row exposes both what was simulated and how it was simulated. The query is filtered to one building so its results can be followed across the scenarios.

Table 18 shows the traced results for one building in Feijenoord. Each row carries the result value together with its unit of measure (`kWh/a`). The `engine_gmlid` is the same `eng_simstadt` in every row, so one `SimulationEngine` was reused for every run, while the `settings_gmlid` changes per scenario. The building appears under the *Future Climate, Refurbishment and Urban Greening* scenarios. The PV Adoption results are stored on the roof surfaces rather than on the building, so they are traced from the surface and not from the building. Table 19 shows this, where the pvPotential on the same roof is seen for two seperate scenarios.

#	building_gmlid	attr_name	attr_value	uom	scenario_gmlid	engine_gmlid	settings_gmlid
1	BAG_0599100000618980	yearlyCoolingDemand	5314.0	kWh/a	scn_future_climate	eng_simstadt	ss_future_climate
2	BAG_0599100000618980	yearlyHeatingDemand	43393.0	kWh/a	scn_future_climate	eng_simstadt	ss_future_climate
3	BAG_0599100000618980	yearlyCoolingDemand	4026.0	kWh/a	scn_refurbishment	eng_simstadt	ss_refurbishment
4	BAG_0599100000618980	yearlyHeatingDemand	36297.0	kWh/a	scn_refurbishment	eng_simstadt	ss_refurbishment
5	BAG_0599100000618980	yearlyCoolingDemand	2691.0	kWh/a	scn_urban_greening	eng_simstadt	ss_urban_greening
6	BAG_0599100000618980	yearlyHeatingDemand	39596.0	kWh/a	scn_urban_greening	eng_simstadt	ss_urban_greening

Table 18: Traceable results for a single building in Feijenoord.

#	feature_gmlid	feature_class	attr_name	attr_value	uom	scenario_gmlid	engine_gmlid	settings_gmlid
1	UUID_c6118334-8482-4b79-b540-4eed7e9e7da7	BuildingRoof Surface	pvPotential	11.35	MWh/a	scn_pv_adoption	eng_simstadt	ss_pv_adoption
2	UUID_c6118334-8482-4b79-b540-4eed7e9e7da7	BuildingRoof Surface	pvPotential	11.25	MWh/a	scn_urban_greening	eng_simstadt	ss_urban_greening

Table 19: Traceable results for a single surface in Feijenoord

5.2.7.4. Results are Distinguishable

Here, the result answers the principle CF3 question: *can the simulation results be distingusihed from each other?*

In this case a sample building filtered by its `gmlid` is shown. It possess the results from all four `Scenario` including the result name.

For each *scenario_results* and *building* pair, heating and cooling appear as seperate attribute_name rows (with own id). They have their own `val_real` and `uom` which is not a single row that encodes both. This makes their difference structural, and aggregating by the `attribute_name` (shown as result_name) column will return different results.

An example of the query results, where for one `building_gmlid` the results for heating and cooling are shown and grouped by `attribute_name` is shown below in Table 20.

#	scenario_results_gmlid	building_gmlid	result_name	n_rows
1	sr_scn_future_climate	BAG_0599100000618980	monthlyCoolingDemand	1
2	sr_scn_refurbishment	BAG_0599100000618980	monthlyCoolingDemand	1
3	sr_scn_urban_greening	BAG_0599100000618980	monthlyCoolingDemand	1
4	sr_scn_future_climate	BAG_0599100000618980	monthlyHeatingDemand	1
5	sr_scn_refurbishment	BAG_0599100000618980	monthlyHeatingDemand	1
6	sr_scn_urban_greening	BAG_0599100000618980	monthlyHeatingDemand	1
7	sr_scn_future_climate	BAG_0599100000618980	yearlyCoolingDemand	1
8	sr_scn_refurbishment	BAG_0599100000618980	yearlyCoolingDemand	1
9	sr_scn_urban_greening	BAG_0599100000618980	yearlyCoolingDemand	1
10	sr_scn_future_climate	BAG_0599100000618980	yearlyHeatingDemand	1
11	sr_scn_refurbishment	BAG_0599100000618980	yearlyHeatingDemand	1
12	sr_scn_urban_greening	BAG_0599100000618980	yearlyHeatingDemand	1

Table 20: Distinguishable results for a single building in Feijenoord.

As a follow up, it should also be shown that a `scn_Scenario` can hold *0 to many* `scn_ScenarioResults`, as per the schema. Therefore, the database was enriched with second set of results for the *Future Climate* scenario, where the `scn_Settings` were changed but the `scn_SimulationEngine` (eng_simstadt) remained the same. In this case, if a second run of the same scenario is produced with a different `SimulationEngine` or a different `Settings` combination, the new run lives in its own `scn_ScenarioResults` row and is sharing `scenario_id` with the first run, but differing on `simulation_engine_id` or `settings_id`, and owning its own `scn_transaction` rows (in case needed, refer back to the final Scenario Module of Scenario ADE beta 3 to see class relationships).

What separates the two runs is no longer the `attribute_name` column (heating vs cooling, as before), but the chain of foreign keys from each `scn_transaction` (its `scn_ScenarioResults`) to the engine and settings that produced it. Walking that chain lets us tell two runs apart even when scenario, building, and metrics are identical. By further but joining `cityobject` two more times (once via `simulation_engine_id`, once via `settings_id`) and adding the resulting gmlids to `GROUP BY` exposes both or all other comparitive runs side-by-side. All results for the same building `BAG...61890` filtered for only the *Future Climate* scenario with the additional run can be seen in Table 21.

#	scenario_results_gmlid	engine_gmlid	settings_gmlid	building_gmlid	result_name	n_rows
1	sr_scn_future_climate	eng_simstadt	ss_future_climate	BAG_0599100000618980	monthlyCoolingDemand	1
2	sr_scn_future_climate_v2	eng_simstadt	ss_future_climate_v2	BAG_0599100000618980	monthlyCoolingDemand	1
3	sr_scn_future_climate	eng_simstadt	ss_future_climate	BAG_0599100000618980	monthlyHeatingDemand	1

#	scenario_results_gmlid	engine_gmlid	settings_gmlid	building_gmlid	result_name	n_rows
4	sr_scn_future_climate_v2	eng_simstadt	ss_future_climate_v2	BAG_0599100000618980	monthlyHeatingDemand	1
5	sr_scn_future_climate	eng_simstadt	ss_future_climate	BAG_0599100000618980	yearlyCoolingDemand	1
6	sr_scn_future_climate_v2	eng_simstadt	ss_future_climate_v2	BAG_0599100000618980	yearlyCoolingDemand	1
7	sr_scn_future_climate	eng_simstadt	ss_future_climate	BAG_0599100000618980	yearlyHeatingDemand	1
8	sr_scn_future_climate_v2	eng_simstadt	ss_future_climate_v2	BAG_0599100000618980	yearlyHeatingDemand	1

Table 21: Distinguishable results for two different simulation engines and settings combinations of the same scenario.

5.2.7.5. Scenario Lineage

The scenario lineage test is used to address the foundational intention (F2) referring to comparible sets of scenarios sharing the same context, as well as branching from shared baselines. This intention translates to the *Scenario Multiverse* we created, shown in the xml encoding in Section 5.2.5.5. The results can answer the following question: *For any scenario, can we retrieve the versions that led to it, and for any version, the scenarios that branch off of it?*

Through the use of a recursive SQL function, we can recurse through the `VersionTransitions`, painting an image of how the scenarios are related to each other in the database. The result table below provides evidence that this intention remains possible in the SQL encoding. The table below (Table 22) shows an example of the query that was run to obtain the lineage. In this case, the query starts from the version `ver_scn_pv_adoption` and recursively searches backwards through the version tree until it reaches the root version `v_baseline`.

#	depth	version_gmlid	version_tag	via_transition_gmlid	transition_type	transition_reason
1	0	ver_scn_pv_adoption	pv-adoption-2050	–	–	–
2	1	ver_scn_future_climate	future-climate-2050	vt_scn_pv_adoption	planned	Feijenoord PV Adoption per-surface PV yield + irradiance
3	2	v_baseline	baseline	vt_scn_future_climate	planned	Feijenoord Future Climate DIN18599 H+C demand under projected climate

Table 22: Recursive version lineage from the scenario chain.

Table 23 is the result of the opposite query, where from a given `version` (in this case the baseline) all branching `Scenario` are identified and shown.

#	depth	version_gmlid	via_transition_gmlid	scenario_gmlid	scenario_project
1	0	v_baseline	–	–	–
2	1	ver_scn_future_climate	vt_scn_future_climate	scn_future_climate	Feijenoord Future Climate DIN18599 H+C demand under projected climate
3	2	ver_scn_pv_adoption	vt_scn_pv_adoption	scn_pv_adoption	Feijenoord PV Adoption per-surface PV yield + irradiance
4	2	ver_scn_refurbishment	vt_scn_refurbishment	scn_refurbishment	Feijenoord Tabula AdvancedRefurbishment surface U-values + post-refurb H/C demand
5	2	ver_scn_urban_greening	vt_scn_urban_greening	scn_urban_greening	Feijenoord Urban Greening trees + post-greening H/C and PV results

Table 23: Branching versions from baseline

5.2.7.6. Context and Strategy Differentiation

The operationalization of the Scenario into storable components included a clear distinct seperation between context and strategy. The following result is to test whether this differentiation also exists in the SQL encoding. Therefore, in line with principle C1, the following question is addressed: *If I removed all intervention data, would the context still make sense on its own and vice versa?*

The context is not held in a single place. The boundary conditions sit as parameters in the `Settings`, while the narrative and the temporal horizon sit on the `Scenario` itself through its `description`, `validFrom` and `validTo`. The strategy sits apart from all of these. It is the `VersionTransition` with the feature and attribute transactions it carries.

Table 24 shows the strategy for the *Refurbishment* scenario. The `VersionTransition` holds the explanation of the change through its `reason` and its `type`. The changes themselves are the transactions it carries, whether structural feature changes or attribute changes. For the Refurbishment scenario these are the U-value updates, and each points at the building as a mean or surface applies to. This reads on its own, separately from what the boundary condition of the scenario actually is, which in our case is the future weather.

#	transition_type	transition_reason	change	attribute	value	uom	target_feature
1	planned	Feijenoord Tabula AdvancedRefurbishment surface U-values + post-refurb H/C demand	replace	meanUValue	0.49	W/(m2*K)	BAG_0599100000618980
2	planned	Feijenoord Tabula AdvancedRefurbishment surface U-values + post-refurb H/C demand	replace	meanUValue	0.62	W/(m2*K)	BAG_0599100000618982
3	planned	Feijenoord Tabula AdvancedRefurbishment surface U-values + post-refurb H/C demand	replace	meanUValue	0.53	W/(m2*K)	BAG_0599100000768134

Table 24: Strategy for the Refurbishment scenario, the `VersionTransition` with its reason, type and the changes it carries.

The specific boundary condition for the scenario can be in this case found in the parameters. Table 25 shows it for the same scenario. The weather sits as a parameter in the `Settings` next to every other input. Each parameter carries a `scope` that marks it as an input, but the `scope` does not mark a parameter as context. So the weather is isolated only by matching its name. Without the name match it is returned amongst all the other inputs, mixed with the simulation configuration and even the refurbishment standard. This is why C1 stays partial in Table 33. A dedicated context scope would let the weather be selected structurally.

#	name	value	scope
1	calc_mode	HEATING_AND_COOLING	input
2	consumer_behaviour	STANDARD	input
3	crs	EPSG:28992	input
4	latitude	51.96; °	input
5	location	Rotterdam	input
6	longitude	4.45; °	input
7	physics_library	Dutch Building Physics Library according to TABULA (http://webtool.building-typology.eu/#bm)	input
8	planningHorizonYear	2050.0	input
9	radiation_model	monthly_meteonorm_Perez	input
10	refurbishmentStandard	TABULA Advanced	input
11	usage_library	Germany (based on UsageLibrary.xlsx, sheet: Germany)	input
12	weather_file	Rotterdam-mon-2050.tmy3	input
13	weather_source	KNMI 2050Hn future climate	input

Table 25: Context of the Refurbishment scenario read from the Settings parameters, the weather boundary condition sitting amongst all other inputs.

Additional Simulation Parameters

As mentioned in Phase 1, Simstadt outputs more parameters than are translated in the requirement tables. Especially for refurbishment, additional parameters previously mentioned include the rate of refurbishment, whether buildings should be prioritized in a specific order, and more… In this case, if additional parameters are taken, they can simply be appended in the `SimulationSettings` as additional `Parameter` holding their own `scope, description, name and value`.

5.2.7.7. Understanding the Scenarios and their temporal horizons

In this case we attempt to offer a result to the question of Principle C2: *Could someone reconstruct the scenario's meaning and timing without reading the raw parameter values?*

As mentioned, every CityObject in the schema (`Scenario` , `Version` , `VersionTransition` , `Settings` , `ScenarioResults`) inherits typed `validFrom` and `validTo` fields frm the "ade-hook" on the `CityObject` class. In this case when enriching the scenarios, the full scenario window (`validFrom` and `validTo`) was placed on the `Scenario` itself, taken to own the period during which the scenario is assumed to hold, while each `Version` carries the `validFrom` of the state it represents (2050 for the future versions, 2025 for the baseline) and although alternative could store on the `VersionTransition` , in this example it carries no temporal window.

Table 26 shows understanding a Scenario and it's Temporal Horizon. The four scenarios with `ScenarioTypeValue` field indicating what `type` of scenario they are, and the plausible narrative sitting as a `gml:description` field. Each row is read on its own and without needing to go look at the any other parameters.

#	scenario	scenario_name	scenario_type	valid_from	valid_to	gmldescription_ renamed_to_narrative
1	scn_future_climate	Feijenoord Future Climate (DB-test 10 buildings)	predictive	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01	Current Feijenoord stock evaluated under the KNMI 2050Hn future climate. No physical intervention is applied; only the climate context changes from the baseline.
2	scn_pv_adoption	Feijenoord PV Adoption (DB-test 10 buildings)	exploratory	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01	Rooftop PV adoption explored for the Feijenoord stock under the inherited KNMI 2050Hn climate; PV potential yield is evaluated per roof surface.
3	scn_refurbishment	Feijenoord Refurbishment (DB-test 10 buildings)	exploratory	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01	Feijenoord stock re-simulated under the inherited KNMI 2050Hn climate after applying TABULA AdvancedRefurbishment U-value upgrades to wall and roof surfaces.
4	scn_urban_greening	Feijenoord Urban Greening (DB-test 10 buildings)	exploratory	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01	Street trees added to the Feijenoord baseline under the inherited KNMI 2050Hn climate, to study their effect on rooftop PV yield and on building heating and cooling demand.

Table 26: Scenario Narrative and Temporal Horizons

Additionally, Table 27 shows the four loaded scenarios against, however this time alongside their respective `Version` . Therefore a user or machine can query all Scenarios occuring during a given timeframe, where timing does not depend on visually reading or parsing through parameter values. Or query for a all the `Version` 's to see the horizon of the physical state the `Scenario` assumes.

#	scenario	version	version_tag	valid_from	valid_to
1	–	v_baseline	baseline	2025-01-01 00:00:00+01	–
2	scn_future_climate	ver_scn_future_climate	future-climate-2050	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01
3	scn_pv_adoption	ver_scn_pv_adoption	pv-adoption-2050	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01
4	scn_refurbishment	ver_scn_refurbishment	refurbishment-2050	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01
5	scn_urban_greening	ver_scn_urban_greening	urban-greening-2050	2050-01-01 00:00:00+01	2055-12-31 23:59:59+01

Table 27: Scenario temporal horizons and Versions

These examples show that the scenario horizon is structurally queryable. The `valid_from` and `valid_to` fields are typed `dateTime` slots, populated through the ADE hook, and reachable with a simple SQL join

and filter. If a scenario is loaded with a different validity, for example a 2020–2025 baseline, the same query will separate the scenarios in time, without any change to the query logic or surrounding scenarios.

5.2.7.8. Input Settings and Output Results are Distinguishable

For this case the question the result addresses belongs to principle C3: *If parameter names were anonymized, could we still tell the difference between a setting and a result?*

Intrinsically, the evidence for this question lies in how the UML was designed and is also visible in the clear separation of `Settings` (which composes `Engine` and `Parameter`) from `Results`, where the latter are materialised through `FeatureTransactions` and `AttributeTransactions`. In the common case, the role of a row is therefore carried by the table it lives in. A `scn_parameter` row belongs to a configuration pillar, a `scn_transaction` row belongs to a result or a change record.

It is however in the schema not illegal for outputs to also be stored as `Parameter` rows inside a `Settings` (for example a simulation output that does not have a feature to attach to). To handle this edge case, `Parameter` can use the optional `scope` attribute with the enumeration `input`, `output`, `other`.

So the architecture answers C3 in the common case, and `scope` answers it in an edge case where the things may be stored differently. Without `scope`, a reproducibility query over a mixed `Settings` would silently pick up output rows, and a user would then have to parse or visually inspect the parameter `name` or `description` strings.

Table 28 shows the parameters of each simulation run, grouped by library, settings and scope. The discriminator is what is structural and therefore to distinguishing inputs from outputs requires no interpretation of parameter names. No parameter happens to be stored with the `scope` set to `output` or `other`, and therefore only returns `input` rows.

#	library_gmlid	settings_gmlid	scope	n_parameters	parameter_names
1	lib_simstadt	ss_future_climate	input	12	{calc_mode,consumer_behaviour,crs,latitude,location,longitude,physics_library,planningHorizonYear,radiation_model,scenarioDescription,usage_library,weather_source}
2	lib_simstadt	ss_pv_adoption	input	12	{calc_mode,consumer_behaviour,crs,latitude,location,longitude,physics_library,planningHorizonYear,radiation_model,scenarioDescription,usage_library,weather_source}
3	lib_simstadt	ss_refurbishment	input	13	{calc_mode,consumer_behaviour,crs,latitude,location,longitude,physics_library,planningHorizonYear,radiation_model,refurbishmentStandard,scenarioDescription,usage_library,weather_source}
4	lib_simstadt	ss_urban_greening	input	13	{calc_mode,consumer_behaviour,crs,interventionType,latitude,location,longitude,physics_library,planningHorizonYear,radiation_model,scenarioDescription,usage_library,weather_source}

Table 28: Parameters per Scenario Simulation Settings, grouped by `scope`.

5.2.7.9. Strategy Pillar Traceability

The transaction table is where the two Strategy pillars are put to the test. It holds the physical interventions and the attribute changes that move a scenario from one version to the next. Both pillars can be checked with one simple query each, run directly against the `scn_transaction` table.

ST1 asks that physical interventions are represented without duplicating base-state geometry, with the question *Does this operation require me to re-encode anything that already exists in the base model?* A `FeatureTransaction` answers this. It stores a change as a delta row that references the features it acts on

by id, through `old_feature_id` and `new_feature_id`. The geometry of those features is never copied into the transaction.

Table 29 shows the result. Each intervention names the existing feature it acts on rather than embedding a copy of it. The unchanged members are carried forward by `clonePredecessor`, so base geometry lives once and is reused.

#	transaction_gmlid	change_type	old_feature	new_feature
1	ft_ug_bm_387353_S	insert	–	bm_387353_S
2	ft_ug_bm_387354_S	insert	–	bm_387354_S
3	ft_ug_bm_387355_K	insert	–	bm_387355_K
4	ft_ug_bm_387363_S	insert	–	bm_387363_S
5	ft_ug_bm_394683_S	insert	–	bm_394683_S

Table 29: `FeatureTransaction` rows, each referencing existing features by id rather than duplicating geometry.

ST2 asks that attribute-level changes point unambiguously to their target, with the question *Could an external tool find the exact attribute targeted using only the schema instance?* An `AttributeTransaction` answers this. It names its attribute and points by `parentFeature` to the exact target feature, so the target is found structurally without parsing any name string. The same `AttributeTransaction` class also stores simulation outputs, so the query keeps only the rows owned by a `VersionTransition`. Those belong to strategy changes and not the results.

Table 30 shows the result. Each refurbishment U-value change lands on one exact building or surface. The target is reached through the structural `parentFeature` link, not by reading a name.

#	target_feature	attribute	value	uom
1	BAG_0599100000618980	meanUValue	0.49	W/(m2*K)
2	BAG_0599100000618982	meanUValue	0.62	W/(m2*K)
3	BAG_0599100000618983	meanUValue	0.42	W/(m2*K)
4	BAG_0599100000618984	meanUValue	0.53	W/(m2*K)
5	BAG_0599100000654922	meanUValue	0.45	W/(m2*K)

Table 30: Prescribed `AttributeTransaction` changes resolved to their exact target feature through `parentFeature`.

5.2.7.10. Comparing Cooling Results between Scenarios

To tie the Scenario ADE back to the case of Rotterdam and the DigiTwin4PEDs project, a last additional query was performed to provide evidence for the Scenario ADE's ability to store and retrieve simulation results and compare the cooling demand of buildings between two scenarios directly using SQL.

Table 31 shows for five buildings a comparison between the *Refurbishment Scenario 2050* and *Future Climate Scenario 2050* of the yearly cooling demand. In a GUI, the application layer could perform such a query returning the delta to the user.

#	target_class	target_gmlid	attr_name	uom	Refurbishment_Scenario_2050	Future_Climate_Scenario_2050	delta	pct_change
1	Building	BAG_0599100000618980	yearlyCoolingDemand	kWh/a	4026	5314	1288	32
2	Building	BAG_0599100000618982	yearlyCoolingDemand	kWh/a	9623	10042	419	4
3	Building	BAG_0599100000618983	yearlyCoolingDemand	kWh/a	2792	3607	815	29
4	Building	BAG_0599100000618984	yearlyCoolingDemand	kWh/a	21220	25322	4102	19
5	Building	BAG_0599100000654922	yearlyCoolingDemand	kWh/a	23625	34791	11166	47

Table 31: Comparing Cooling Demand of Building between *Refurbishment Scenario 2050* and *Future Climate Scenario 2050*

5.2.7.11. Full Scenario Multiverse Database

Lastly, as done in the XML encoding the entire *Scenario Multiverse* was loaded into the 3DCityDB. Table 32 shows the database statistics.

#	objectclass_id	class_name	n_rows
1	20100	Version	5
2	20110	VersionTransition	4
3	20120	FeatureTransaction	1599
4	20130	AttributeTransaction	30763
5	20200	SimulationLibrary	1
6	20210	SimulationEngine	1
7	20220	Settings	4
8	20300	Scenario	4
9	20310	ScenarioResults	4

Table 32: Statistics for Feijenoord Neighborhood Scenario Multiverse

The full database report can be seen below in Figure 5.15 :

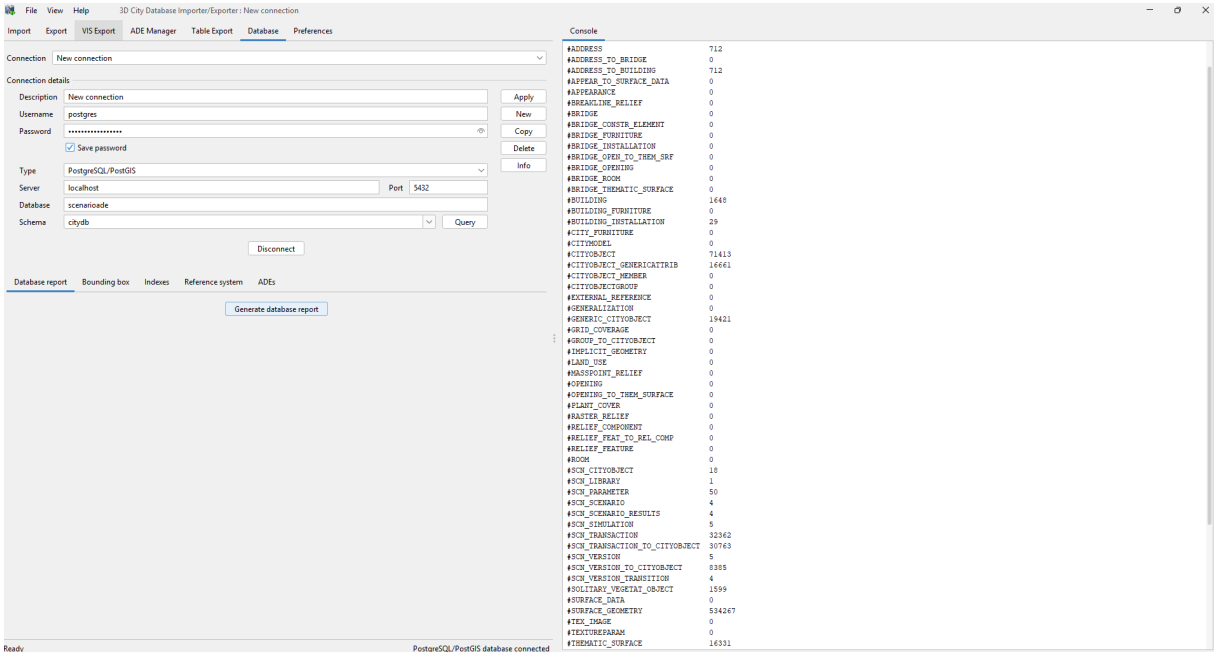


Figure 5.15: Full Importer/Exporter Database Report for Scenario Multiverse

5.2.8. Scenario ADE Final Schema

The iterative refinement reached its final state with Beta 6, where the schema had to be frozen to allow time to start testing and encoding. The XSD used for the encoding and database tests in Section 4.4 is the v3 release of the Scenario ADE. Table 33 summarises the status of each design principle from Section 3.3.3 across the three evaluated betas. The status reflects how well each schema performed in respect to the principle. The Status is not derived from strict conformance on how much parameters could be mapped or how many validation gaps existed, it is instead interpreted from the experience and insights gained from the iterative methodology and respective encodings. A *unmet* label corresponds to a gap, where the schema could neither conceptually or structurally support the principles. *Partial* corresponds with the schema having the ability to either conceptually or structurally support the principle, however the principle cannot yet be explicitely met. *Met* corresponds to the schema fully supporting the principle and lastly, deferred is used when a beta did not address the gap.

A brief reason for the final verdict is given below per each foundational intention and design principle:

- F1 is met. The schema is domain agnostic and does not rely on any EnergyADE or vocabulary specific to any other domain.

- F2 is met. Scenario set conherence is achieved through versioning lineage. Scenario can be created in comparable sets stemming from the same baseline version.
- F3 is met. The schema keeps forward compatibility in mind. It can migrate to CityGML 3.0 with minor changes. Migrating would in fact even resolve some of the current limitations.
- C1 stays partial. The weather file that sets the boundary condition sits as a `Parameter` in `Setting` s next to ordinary inputs, so it is retrievable by naming rather than structurally. The proposed context scope would close this.
- C2 is partial. The horizon sits on the `Version` as `validFrom` and `validTo`, and the narrative travels with the scenario as a per change reason on `VersionTransition.reason`. The narrative storyline should however be more explicit on the scenario, which is why this is partial. The discussion elaborates on this.
- C3 is met. `Parameter.scope` marks each value as input, output or other, so its role is explicit.
- ST1 is met. Interventions are stored as `transactions` and `clonePredecessor` carries unchanged members forward, so attributes, children and geometries are never duplicated.
- ST2 stays partial. `AttributeTransaction.parentFeature` points by `xlink` to the exact target feature. However in cases where the base model contains `citygmlGenericAttributes` that need replacing, their exact name is needed to be known to either `delete` or `replace`. This is not ideal especially if a baseline citymodel relies heavily on generic attributes which need targetting or changing as part of a strategy or result. This is a limitation of generic attributes and is further elaborated upon in the discussion.
- CF1 is met. `SimulationEngine` paired with `Settings` forms a separable run that can be reused across scenarios.
- CF2 is met. `ScenarioResults` stores each result with a `parentFeature` link back to the object and records the `engine` and s`ettings` that produced it.
- CF3 is met. `Results` travel through the typed `AttributeValue` union inside `ScenarioResults`, so several result types for one object stay distinguishable. However, it is worth noting that as `attributeName` is a unconstrained string, so the schema would not be able to tell apart two transactions that target the same `parentFeature` if they had the exact same name.

ID	Principle	Beta 3	Beta 5	Beta 6
F1	Domain Agnosticism	unmet	met	met
F2	Scenario Set Coherence	partial	unmet	met
F3	Forward Compatibility	unmet	met	met
C1	Boundary conditions and interventions remain independently retrievable.	unmet	partial	partial
C2	Temporal horizons and narratives travel with the parameters.	partial	partial	partial
C3	Parameters carry metadata to identify their role (input vs. result).	partial	partial	met
ST1	Physical interventions are represented without duplicating base-state geometry.	partial	deferred	met
ST2	Attribute-level changes point unambiguously to their target.	met	deferred	partial
CF1	Simulation settings are separable, reusable entities.	unmet	partial	met
CF2	Results are traceable to the specific object and simulation run.	unmet	unmet	met
CF3	Multiple result types for the same object are structurally distinguishable.	unmet	unmet	met

Table 33: Status of each design principle across the three refined betas. *partial* marks structural support that is not enforced, and *deferred* marks a principle deliberately set aside for a later iteration.

6. Discussion

This chapter reflects on the development of the Scenario ADE. It interprets the simulation and validation results, discusses how the schema addresses the research gap and relates to established provenance standards, and sets out the limitations and recommendations for future refinement.

6.1. Key Findings

This thesis set out to test and refine the Scenario ADE so that a CityGML 2.0 dataset can hold several alternative futures of the same neighbourhood and tie each one to the simulation that produced it. The gap it addresses is twofold (Section 2.4.5). Standard CityGML 2.0 represents a single city state and has no way to manage concurrent hypothetical alternatives in one dataset. The artifact (Scenario ADE Beta 3) can attach simulated values to objects but do not bind them to the run, engine and parameters that produced them. None of the existing mechanisms, the CityGML 3.0 versioning module, the Dynamizer, Git-based versioning, or provenance standards can specifically link what changed to why it changed and under which assumptions.

The simulation runs informed which data the schema must hold. Conducting the simulations and observing the results motivated four requirements. Simulation inputs must be stored and linked, enrichment data must be traceable to the features it describes, every result must be linked to what produced it, and results must be kept at their native resolution.

A scenario definition, informed by literature and the empirical findings, was decomposed into the three pillars of Context, Strategy and Configuration. These pillars and the simulation runs motivated the intentions and design principles that guided the refinement from Beta 3 to a final Beta 6, which backports the CityGML 3.0 versioning module among other improvements. Beta 6 was validated in two encodings. Nine foundational operations and the four test scenarios were encoded for Feijenoord as CityGML XML and all validated, then loaded into a 3DCityDB instance where lineage and cross-scenario comparison are queryable. Of the eleven design principles, most are met and three, C1, C2 and ST2, remain partial (Table 33).

6.2. Interpretation

6.2.1. Sensitivity of the Energy Simulations

From the simulation runs conducted in the first phase, there are two results matter the most for the design. An invisible configuration choice moves the outputs, since the radiation model alone separates the demands, with Perez computing more solar gain than Hay (Section 5.1). The WWR has a larger and more uneven effect. A uniform 20% archetype increases the measured cooling demand by 57.7% in Feijenoord and 61.8% in Prinsenland, while heating is far less sensitive, so an assumed neighbourhood-wide WWR is less valuable than per-building or per-surface measured values. Both cases make the same point for the schema. A result cannot be read without the input and the feature behind it, which is the empirical case for storing inputs and binding every result to the run that produced it. The findings are interesting in their own right, but the lesson that carries into the design is how such simulations should be stored. Keeping each result with its full provenance is what gives decision makers easier and more comparable access to them.

6.2.2. From Scenario Theory to Scenario Data

The Scenario ADE is grounded in the XSP separation of contextual and transactional environments (Abou Jaoude et al., 2022) and in Spaniol & Rowland (2019)'s synthesis of what a scenario is within the intuitive logics tradition. Operationalising that definition into concrete data requirements forces decisions a textual definition never has to make. Turning it into pillars made implicit assumptions explicit, and a different reading of the literature or different scenario data could have produced different principles and a different schema.

Spaniol's "future purview" characteristic is one example of where this occurs. Taken literally a baseline or a historical state would not qualify as a scenario, which is awkward when the ADE builds scenarios from `Version`s that are often baselines, and it also excludes the backcasting and forecasting scenarios widely used in urban planning that Börjeson et al. (2006) distinguishes. The thesis instead reads "future-oriented" as a property of the scenario set rather than of each version. A scenario compares alternative futures while its versions can be past, present or future, and backcasting fits this naturally. A practical schema stores data for users who apply it outside strict theoretical definitions, so a literal interpretation would only have reduced its utility.

6.2.3. Conflation of Context and Configuration

The three pillars treat Context and Configuration as distinct (Section 3.3.1), but the final schema does not separate them cleanly. The backported versioning module gives Strategy and Configuration a structural home but has no construct for context (Section 4.3), so context is scattered across slots, the horizon on `validFrom` and `validTo`, the narrative on `gml:description`, and the weather file as a `Parameter` in `Settings`. The KNMI 2050 file is a boundary condition of the scenario yet is stored as an ordinary `input` next to the simulation parameters, distinguishable only by its name (Section 5.2.7.6). This is why C1 stays partial (Table 33). It follows from building on a versioning structure that knows states and changes but not the boundary conditions inherent to scenario planning. A dedicated context scope would fix it (Section 6.5).

6.2.4. Domain Agnosticism versus Semantic Rigidity

The schema was first conceived as an extension of the Energy ADE with energy native classes. It then shifted to an agnostic layer that sits on any city model and manages only the deltas, which made it reusable but without pushed attribute modifications such as energy demand and PV potential through loosely typed generics. The same yearly cooling demand was encoded three ways, as a CityGML generic attribute, a Scenario ADE `simpleAttribute`, and a typed `nrg3:Energy` feature. What changes across them is how much the value explains itself. The generic forms carry a number and a name, while the Energy ADE form carries the end use, carrier and unit in typed fields a tool can read without at all relying on the convention of the input user. The parameters are already constrained. Domain agnosticism allows for the reach to be appliclable anywhere, but pays for it with more limited self description. When a domain ADE such as the Energy ADE is in use, the most precise encoding available should be preferred, here the typed `nrg3:Energy` feature. The more generic encodings then serve as fallbacks of increasing generality, a `simpleAttribute` for a property the schema already defines, and a CityGML generic attribute where nothing more specific exists. Precision is used wherever the domain allows it, and generality only where it does not.

The clearest cost of the domain agnostic decision are limitations such as `AttributeTransaction.attributeName` requiring to be a unconstrained `xsd:string`. Promoting it to a `gml:CodeType` would make results machine readable but would fix the vocabulary to a closed list. This is a tradeoff between flexibility and semantic rigidity, and the choice was to keep it open. A lighter alternative to a closed codelist would be to add an `attributeNameSpace`

field beside the `attributeName`, which lets a tool check the named attribute against the data model it points to rather than against a fixed list. This refinement is set out in the residual schema gaps (Section 6.4.2). Ultimately, whether a codelist helps or hurts the schema requires testing on other domains as well (Section 6.4).

6.3. Implications

6.3.1. Addressing the Gap

The schema addresses the research gap identified in the related work (Section 2.4.5). The CityGML 3.0 versioning module manages parallel alternatives at the feature level but does not link a change to the simulation that produced it, or operate at the attribute level. The Dynamizer animates a single attribute over time but cannot branch into alternatives or hold scenario-specific results. Git and GeoGig versioning track file-level deltas but are blind to the semantic structure of a scenario and to its inputs and outputs. None of them links what changed to why it changed. The Scenario ADE adds that link. The transactions carry what changed, the `VersionTransition` and `Scenario` carry why, and `ScenarioResults` binds each result to the engine and settings that produced it.

The contribution is therefore not a new versioning mechanism but the binding that happens around one. The versioning logic is repurposed in two ways CityGML 3.0 does not use. It stores what-if data at both feature and attribute level inside the Scenario ADE container on top of the city model, imitating the feature transaction with an added `parentFeature` for attribute-level changes, and it acts as a provenance tracker that links each result back to the exact resolution it was produced at.

Wrapped in the Context, Strategy and Configuration framing, the schema keeps scenarios in comparable sets that branch from a shared baseline (F2) and ties every result to the feature and run behind it (CF2). The database shows both, where lineage can be recovered from any scenario and cooling demand compared across scenarios in a single query (Section 5.2.7). The four “what-if” simulation runs already demonstrate the value before any schema is involved. Each branches from the same baseline and moves demand in its own direction, the future climate raising cooling and lowering heating, the refurbishment lowering both, and the urban greening inverting cooling while trading away some rooftop PV. They only carry meaning when set against each other and the shared baseline, which is exactly the relation the schema is built to hold. This is what makes the alternatives usable for decision making in projects such as DigiTwins4PEDS.

A further benefit is that CityGML 2.0 never had a way to track real-world feature changes. As a byproduct, the Scenario ADE can also serve as a versioning tool for real-world changes within a CityGML 2.0 city model.

6.3.2. Borrowing a Provenance Standard

The Configuration pillar arrived at a lightweight custom form of provenance, tying a result to the engine, settings and inputs that produced it. This overlaps with the two standards reviewed in Section 2.4.4. In ISO 19115 (ISO, 2014) the run is a process step, the `SimulationEngine` its processor, the `Settings` its parameters and a result a source output. In W3C PROV (Belhajjame et al., 2013) the same pieces are an activity, an agent and a generated entity, and the `input` and `output` values of `Parameter.scope` are a distinction both standards already draw. Three directions could have shaped this part of the schema differently.

Adopting a standard in full was one option. The Metadata ADE already brings ISO 19115 into CityGML (Labetski et al., 2018), but it describes how input data was created rather than carrying simulation outputs, and a full PROV graph or ISO 19139 record would add complexity that is most likely overkill for a simple binding of results to runs. Neither standard models a state or a change between states, so the versioning core would remain regardless.

A second option was to design the pillar to be context-independent from the start. Grounding it in one domain with concrete requirements is what gave it its usefulness, so a standard first design might have been more interoperable but would have lacked the grounding the context provided it. This is a similar deduction to the point made to not begin the development of the schema from the testADE, and instead ground it in a domain specific context first.

The most realistic direction however is to borrow some of the core vocabulary to allow non simulation scenarios to also have the same level binding. Here a example of using more general concepts and vocabulary from the W3C PROV is presented. The schema records the provenance of computed results well but has no structured way to record deliberate strategy changes, where a planner or municipality acts with no engine and no settings. Abstracting `ScenarioResults` into PROV-DM concepts could be an approach that resolves this. The `SimulationEngine` generalises to an *Agent*, `ScenarioResults` to an *Activity*, and the `Settings` to a *Plan*, so the plan can hold parameters, a policy document or anything else the agent used. In practice the `SimulationLibrary` becomes an `ActivityLibrary` whose classes use CodeLists to state their nature, letting one library serve both a software engine and a human planner. A `VersionTransition`, which now carries only a `reason`, could then instead, through the `Scenario` have the provenance to what created it and under what assumptions just like simulation results through the Activity, Agent and Plan. For example, a exploratory zoning scenario the transition would record Gemeente Amsterdam as the Agent and its policy document as the Plan (or as a parameter sitting on a plan class), instead of forcing that onto `Scenario` or `Version` as generic attributes and recreating similar conflation to what Beta 3 initial had. The need to support more generalised concepts to allow for not only the simulation runs is a clear limitation. This may be one approach.

This keeps the same provenance logic but generalises its vocabulary, so it is carried forward as a recommendation (Section 4.3).

6.3.3. CityJSON Encoding

The schema was encoded in CityGML XML and in a 3DCityDB relational schema, and third and missing encoding is CityJSON. It is the natural next encoding to benchmark against the XML and SQL ones, and it is carried forward as a recommendation (Section 6.5). It was however not pursued yet in this thesis for three main reasons.

The first is the simple natural progression and order of work. CityJSON encodes the CityGML 3.0 data model, so a Scenario ADE encoding in CityJSON only makes sense once the schema itself has moved to CityGML 3.0. Beta 6 still validates against CityGML 2.0 and backports the versioning module into that environment (Table 10). Therefore, the migration to CityGML 3.0 is the prerequisite step, and a CityJSON encoding follows it.

The second is the tools and workflows the urban energy domain used in the DigiTwin4PEDs and prior work depends on. The energy modelling workflows adapted to collect empirical evidence all are in CityGML 2.0. SimStadt reads CityGML 2.0, not supporting CityGML 3.0 and produces the simulation outputs that ground every requirement in this thesis. Another UBEM tool, CitySIM also runs on CityGML files. A CityJSON encoding would therefore not connect with

the existing domain workflows, so work beginning with CityJSON would have to be grounded differently. This could be achieved if the thesis grounded itself in a different context.

The third reason comes from the current CityJSON schema itself. At the time of writing this thesis, CityJSON release 2.0.2 is out and as mentioned it already implements the CityGML 3.0 data model. The new interior building classes that CityGML 3.0 introduced are present, such as `BuildingRoom`, `BuildingStorey`, `BuildingUnit`, `BuildingConstructiveElement` and `BuildingFurniture`, alongside `OtherConstruction` and `Waterway`. What the schema does not contain is the versioning module, which the Scenario ADE heavily relies on. There is no `Version`, no `VersionTransition` and no transaction construct anywhere, there is no Dynamizer or time series construct either. Instead it relies on its git-like versioning, a path which the Scenario ADE would require exploring during the encoding. Generally, for these reasons the CityJSON encoding would require the most alterations out of the three and would not have been achievable within the scope of this master thesis. One interesting finding worth noting is that CityJSON encoding incorporates a metadata structure based on the ISO 19115 standard. This further validates exploring whether the Scenario ADE should adopt provenance which aligns more with a standard.

Lastly, CityJSON replaces CityGML ADEs with what is called Extensions, which are JSON files that add city objects, attributes and root properties to the core model (Ledoux et al., 2019). The Scenario ADE would therefore have to be re-expressed as a CityJSON Extension, and this is where versioning classes it relies on would need to be defined inside that Extension rather than reused from the core. Extensions are also less flexible than ADEs, since JSON Schema supports neither inheritance nor namespaces. A CityJSON encoding is thus a possible future step, but only after the move to CityGML 3.0 and once the CityJSON limitations in terms of data modelling for Extensions have been accurately evaluated with respect to the Scenario ADE.

6.4. Limitations

There are three kinds of limitations present, ordered by how much each one restricts the contribution. The first is the scope of the validation. It sets how far the claims can reasonably be generalised. The second is what the schema still cannot express through its own structure. Most of this rests on convention and would be resolved by a move to CityGML 3.0. The third is the tooling that does not yet exist around the schema and is mentioned inbetween residual schema gap arguments.

6.4.1. Scope of the Validation

Every case study comes from the energy domain, and the four test scenarios form a small tree, one branch from the baseline and three from that branch. Domain agnosticism (F1) emerged during the work rather than being planned. Both the mapping and the validation assessments allowed the non conforming classes to naturally be quite easily identified however as the schema on the structure of the schema and has not been demonstrated on a second or third domain. The schema has not yet been tried outside the narrow domain that shaped it, so where it would hold and where it would fail is still unknown. Testing it on another domain is therefore carried forward as a recommendation for future work.

6.4.2. Residual Schema Gaps

Three of the design principles are only partly met, C1, C2 and ST2 (Table 33). In each case it is because the property holds by convention rather than by structure. The weather file that fixes the boundary condition (C1) is stored as an ordinary `Parameter` in `Settings`, so it can

be found by name but nothing marks it as a boundary. The scenario narrative (C2) has no field of its own and is written into the inherited `gml:description`. ST2 is the most concrete. The `citygmlGenericAttributes` are DataTypes and carry no `gml:id`, so when one must change as part of a strategy a `parentFeature` xlink cannot reach it cleanly. The transaction has to be given the attribute name in advance, and since that name is a free string, two similar names could end up describing the same thing. Identical names cannot collide on a single object, since CityGML does not allow two generic attributes with the same name on the same city object. The remaining gap is that a name on its own does not say where the attribute is defined. A useful improvement would be to pair the `attributeName` with an `attributeNameSpace` field. A simple attribute could then be bound to the schema that defines it, for example `core:relativeToTerrain` or `bldg:yearOfConstruction`, while a generic attribute would carry no namespace. The consumer could then check where each attribute comes from. Sorting this out otherwise falls to an application layer rather than the schema. The XPath construct explored earlier (Section 4.3.1.2) might also remove the ambiguity, and in any case it mainly matters for datasets that keep everything in generic attributes rather than using a dedicated ADE or the standard attributes defined in CityGML core and its other modules.

Leaving `attributeName` unconstrained was a deliberate choice, since the open field is what makes the schema flexible and the conflation above is the cost of that. Two further gaps sit beside it. A real-world change and a scenario-driven change look identical in the structure, so they can only be separated by tracing back to the scenario. There is also no way to merge scenarios or resolve conflicts, because only diverging branches were tested. Although the logic would most likely be taken up by a application layer, scenarios which merge could have also been tested.

A final pair of gaps is about granularity rather than what a transaction can record. A replace is clean when the target has a single `gml:id`, as a whole building does. Operation 6 showed that not all objects are addressed by a single `gml:id`. For the 3D BAG roof, each detected plane gets its own `bldg:RoofSurface` and `gml:id`, so a roof of N planes takes N deletes and one insert. A roof modelled as one `gml:Multisurface` constructed from a single polygon would have a single `gml:id` and replace cleanly, but 3DBAG reconstructs roofs as multiple polygons. This has to do with the CityGML geometry module and how geometries can be constructed from their primitives. As there exists different approaches to represent geometries, starting from LOD2 onwards. As the LOD increases, so would the possibility of coming accross similar issues does. Therefore a best pratice and simpler route is probably to replace the whole building instead. What matters is therefore not whether a change is geometric but whether the target is a single object.

6.5. Recommendations

The iterative refinement and evaluation of the three Scenario ADE betas surfaced a number of alternatives and limitations. The following schema refinements and directions for further work draw them together.

6.5.1. Schema Refinements

1. The `ScenarioTypeValue` codelist was left open during testing so it could be filled from experience. The urban planning literature gives a foundation, with Börjeson et al. (2006)'s division of forecasting, backcasting and exploratory planning, and a baseline type added for the reference state that makes scenario sets comparable. The recommended values are

    - `baseline` for the reference state from which other *Scenarios* branch.

- `explorative` for what-if *Scenarios* that explore what can happen under varied context or strategies.
- `predictive` for forecasting *Scenarios* that describe what is expected to happen.
- `normative` for backcasting *Scenarios* that describe how to reach a target state.

2. A `context` value should be added to `Parameter.scope` so the weather file can be retrieved as a boundary condition rather than an ordinary input, moving C1 closer to its intention. This also supports any exploratory scenario that needs its context separated from its configuration.

3. An optional `narrative` [0..1] slot should be added to `Scenario`. Explorative scenarios make up most of those tested and all should carry a plausible storyline, which is currently placed onto the inherited `gml:description`. A dedicated slot would move C2 towards met.

4. Scenario-generated features should be carried inline as children of the `VersionTransition`, inside the `FeatureTransaction` that introduces them, rather than placed in the city model and linked back to the scenario. This is the approach the encodings now take, and it keeps such features out of the baseline. Containment is what tells a scenario-driven feature apart from a real-world one, since the feature belongs to the transition that created it and its provenance is read from where it sits. An earlier idea was to add a `generatedByScenario` back-reference on the feature, but this does not hold. The same new feature could be introduced by more than one scenario, so a single reference is not enough. It would need a multiplicity of 0 to many and would become a many-to-many relation. Carrying the feature inside its transition avoids this, since each scenario holds its own instance and the structure stays cleaner. This also lets an application filter the real-world city model for visual or analysis purposes by reading only the root features, without recursing through every `VersionTransition`. The approach mirrors the `Version.tag` used in the encodings to mark a `Version` as `baseline` or `scenario`, which made the branching baseline explicit and improved scenario set comparability. Comparability itself does not depend on whether scenarios sit as a flat set off the baseline or as a chain branching from one another. It depends on sharing a common ancestor version, which the `VersionTransition` graph encodes and the recursive lineage query recovers. A shared assumption is therefore best modelled as a shared ancestor that the branches inherit, which also avoids duplicating the feature. The convention that follows is that cumulative scenarios chain while mutually exclusive ones branch as siblings, since a chain carries the predecessor's changes forward through `clonePredecessor`. The one case the model cannot express is a feature shared across branches that have no common ancestor holding it. That would require a merge, which is the same boundary as the absence of conflict resolution noted above. An optional `provenance` marker with values such as `real` or `fictive` could still be added for features that remain in the city model, and the containment already closes the one-directional asymmetry the back-reference was meant to solve.

5. Generalise the provenance tracking to cover both computational runs and deliberate strategy changes by making the Configuration pillar more abstract (Section 6.3.2). Aligning with W3C PROV, the `ScenarioResults`, `SimulationEngine`, `Settings` and `SimulationLibrary` classes broaden into an *Activity*, *Agent*, *Plan* and `ActivityLibrary`. For a concrete example, a simulation run records the SimStadt engine as the Agent and the KNMI 2050 weather and parameters as the Plan. A deliberate zoning change, where there is no engine and no settings, records the municipality such as Gemeente Amsterdam as the Agent and its policy document as the Plan. Both then carry the same Activity, Agent

and Plan binding, so a planner-driven change becomes as traceable as a simulation result rather than being forced onto a `Scenario` or `Version` as generic attributes.

These refinements will move forward as future work, introduced after the conclusion.

7. Conclusion

Cities account for most of the world's energy use and emissions, and a warming climate is shifting Dutch neighbourhoods from heating to cooling dominated demand. Meeting the Dutch 2050 targets and reaching Positive Energy Districts depends on decisions at the neighbourhood level, where planners will weigh strategies such as refurbishment, PV adoption and urban greening against one another. Each is a different future of the same neighbourhood, and a decision maker needs to compare them side by side and trust the assumptions, figures or numbers behind each one of them. Standard CityGML 2.0 cannot hold these alternative futures in one dataset, nor tie each result to the run that produced it. Closing that gap, in the context of the DigiTwins4PEDs project and the Dutch energy transition, is what this thesis set out to do.

This thesis introduces a new version of the Scenario ADE for CityGML 2.0, derived through an iterative refinement process focusing on empirical evidence derived from simulations, scenario theory and requirements. The work applies Exploratory Scenario Planning (XSP), grounded in the Intuitive logics tradition synthesis of scenario characteristics and the urban planning scenario literature, and finalises in the implementation of the final schema with its XSD and DDL scripts. The final schema was then validated through case study scenario's, first proving they could support fundamental operations and afterwards neighborhood scale encodings. The SQL encoding was additionally tested through proving the Scenario ADE intentions structurally match and that what is represented in the schema holds true in the database. Through the research process of the thesis and the insights and findings gained at each stage of the Scenario ADEs design, the research and sub-questions can be answered. First the three sub-questions are answered consituting the base required for answer the research question which follows.

7.1. Research Questions

7.1.1. Sub-Question 1

What entities, properties and relations must a Scenario ADE represent to model an urban scenario?

The scenario is decomposed into three pillars: Context, Strategy and Configuration, grounded in the scenario definition adopted in the related work (Figure 3.3) and through evidence gathered by running energy simulations. The Context holds the external boundary conditions a scenario assumes, the Strategy holds the interventions applied inside that context, and the Configuration holds the simulation engine, its settings and its results. The Context and Strategy split comes from the exploratory scenario planning separation of contextual and transactional environments, and the Configuration pillar was added by this thesis to make the simulation experiments reproducible. Each pillar maps onto schema entities, properties and relations. The decomposition fixed the roles these entities must fill, while the concrete classes that realise them were settled through the refinement in the next sub-question. The `Scenario` is the root container for the whole scenario. Strategy is realised as a `VersionTransition` carrying `FeatureTransaction` and `AttributeTransaction` deltas, Configuration as a `ScenarioResults` block tied to a `SimulationEngine` and `Settings`, and Context as metadata across the `Scenario`, the `Version` and the `Settings` parameters. The decomposition was turned into a set of design principles that guided and evaluated the schema (Table 1).

The simulation runs inspired the need for a configuration pillar but also grounded which concrete data each pillar must hold. The Phase 1 runs showed that a single configuration choice affects the outputs, since the radiation model alone separates heating and cooling demand (Figure 5.2), and that one attribute of one building matters, since a standardized archetype 20% WWR relatively way overpredicts measured cooling demand (+57.7%) in Feijenoord and in Prinsenland (+61.8%) (Table 12, Table 13). The effect is unevenly spread across individual buildings, as can be seen in the outliers (Figure 5.6). These empirical observations gave four data requirements, that simulation inputs must be stored by the schema, that enrichment must be traceable to the feature it describes, that every result must be linked to what produced it, and that results must be kept at their native resolution. Comparison is what makes the decomposition useful. The four what-if runs each branch from one baseline and only carry meaning when set against each other and that baseline (Figure 5.7). This is why the structure supports keeping scenarios in related sets rather than as isolated states (although possible to support one off scenarios as well). These empirical findings were summarised by the Phase 1 package diagram (Figure 4.6).

7.1.2. Sub-Question 2

How can an existing Scenario ADE schema be iteratively refined and evaluated against design principles derived from scenario theory and empirical evidence?

The refinement followed a Design Science Research methodology. The empirical findings translated into the concrete data requirements forming the relevance cycle. The decomposition to scenario pillars forms the rigor cycle. Lastly, the beta 3 is the artefact from which the iterative design cycle of the Scenario ADE starts. Each version was evaluated against the design principles and the four test scenarios through semantic mapping and a validation assessment, then refined and re-evaluated (Figure 3.4, Figure 3.5). Three versions were evaluated, Beta 3, Beta 5 and Beta 6, since Beta 4 was an interim project version.

Beta 3 was simple and domain dependent. It collapsed context, interventions and results into one untyped genericParameter slot, depended on Energy ADE classes, and expressed lineage only through a flat `dependsOn` link (Figure 4.7). Its evaluation produced the first set of recommendations (Table 8), feeding into the transition towards Beta 5.

Beta 5 made the schema domain agnostic but exposed new gaps. It removed the Energy ADE modification classes and introduced the `SimulationLibrary` and `Simulation` to separate runs from scenarios. In doing so it lost a home for attribute changes, lost scenario lineage, and still could not trace a result to its run (Figure 4.10, Table 9).

Beta 6 closed the open gaps and solving the lineage and results provenance gaps by backporting and slightly customizing the CityGML 3.0 versioning module. The mapping between the two is given in Table 10. The `Version`, `VersionTransition`, `FeatureTransaction` and `AttributeTransaction` classes gave Strategy a structural home, `AttributeTransaction` let a single attribute change without duplicating geometry (Table 35), `VersionTransition` restored lineage between scenarios, and `ScenarioResults` bound each result to the engine and settings that produced it (Figure 4.14).

The principle status improved across the iterations. This shows the schema moving from mostly unmet in Beta 3 to eight of eleven principles met in Beta 6, with only C1, C2 and ST2 left partial (Table 33).

7.1.3. Sub-Question 3

To what extent can the operationalised Scenario ADE, in its physical encodings, support the management, analysis and comparison of urban energy scenarios?

To a large extent the operationalised schema supports the management, analysis and comparison it was built for, demonstrated across two physical encodings, XML and SQL. The Beta 6 UML was mapped to a CityGML XML application schema and to a 3DCityDB relational schema, following the Model-Driven Approach (Bachert et al., 2024) applied once the iterative refinement process was completed. A shared set of Python scripts then moves data from the raw SimStadt outputs, through cleaned per building and surface tables, into the Scenario ADE constructs of both encodings (Figure 4.17).

The nine foundational operations of the XML encoding each pass XSD validation (Figure 5.8), and the four case study scenarios are encoded for Feijenoord and then grouped into one shared Scenario Multiverse file. Storing them together rather than as four separate files is 28.7% smaller (Table 15).

The database encoding makes management and comparison queryable. A `SimulationEngine` and its `Settings` are reused across scenarios rather than duplicated (Table 17), each result is traced back to the building and the run that produced it (Table 18), and several results on one object stay distinguishable (Table 20).

Scenario lineage and cross scenario comparison both work in SQL. A recursive query recovers the versions that lead to any scenario, and the scenarios that branch from any version (Table 22, Table 23). Cooling demand can be compared directly between the Refurbishment and Future Climate scenarios in a single query (Table 31), which is the comparison the schema was built to support.

Interaction is currently script driven. The XML and SQL workflows are handled by custom scripts, and no application layer or GUI yet exists for an end user to query, merge or extract scenarios.

7.1.4. Research Question

How can a Scenario ADE for CityGML 2.0 be developed and implemented to support the structured management, analysis, and comparison of multi-temporal scenarios in urban applications, using urban energy as the case study?

An explicit answer to this question can be answered through the final Scenario ADE Beta 6 schema and it's functionality. It is a working schema reached through three iterations starting from the Beta 3 artifact. Its four modules give the Beta 6 schema its final shape, these being the core, the scenario, the versioning and the simulation library (Figure 4.13, Figure 4.14, Figure 4.12, Figure 4.15).

The data model is grounded in the decomposition of scenario into three pillars: Context, Strategy and Configuration (Figure 3.3). These pillars are derived from the Exploratory Scenario Planning (XSP), grounded in the Intuitive logics tradition synthesis of scenario characteristics (Spaniol & Rowland, 2019) and the urban planning scenario literature (Abou Jaoude et al., 2022; Börjeson et al., 2006). To give Strategy and Configuration a structural home the schema reaches and takes inspiration from CityGML 3.0. It borrows the CityGML 3.0 versioning module and backports it, so the result still validates against CityGML 2.0 and stays usable in the current tooling (Table 10).

Whether it works was tested through intial foundational operations and at neighbourhood scale in two encodings. The nine foundational operations each pass XSD validation (Figure 5.8), and the four Feijenoord scenarios load cleanly into a 3DCityDB instance. Held together as one shared multiverse, and encoded through the same way, the four are 28.7% smaller than separate files (Table 15), and they branch from a single baseline into one tree (Figure 4.18). Eight of the eleven design principles are met, with C1, C2 and ST2 left partial (Table 33). The schema can hold several alternative futures of one neighbourhood in a single dataset and tie each result back to the run that produced it.

7.2. Future Work

The recommendations in Section 6.5 set out the schema refinements which still needed addressing. These refinements, along with the other discussion points can be concluded into five directions of future work.

1. All recommended refinements should be considered and applied to the next iteration of the Scenario ADE. The methodology culminated in a schema proven to meet almost all the initial intentions and design principles on a conceptual level, as well as concretely meet all the data requirements. The encodings produced were also all valid. That being said, areas where no dedicated slot could improve explicit representation of scenario parameters or over reliance on generics is present which could be improved by many of the refinements. Small changes such as implementing a narrative slot or a added codelist value for the context parameter and formalising the `ScenarioTypeValue` codelist are all easily applied changes which do not effect the overall structure and compliance of the schema with the intentions. The two last refinements: asymmetric relationships back to scenarios and generalising the provenance tracking are larger refinements which could affect the functionality and should be tested before being applied.

2. The schema has been developed and tested only within one narrow scope, urban energy modelling derived from simulation runs using Simstadt. Testing the schema on a non-energy domain and on non-simulation scenarios such as flood risk, mobility or policy related scenarios would be required. This is what would actually test the domain agnosticism intention and show whether open or typed attributes suit other domains, and whether the PROV vocabulary recommendation would significantly improve operationality.

3. The schema should align with the CityGML 3.0 Versioning module rather than backporting it into CityGML 2.0, which would resolve some of the limitations above and remove the need for the backport. This is not a clean revert though. The pragmatic choice in Beta 6 was to base the transaction classes on `core:_CityObject`, which gave them a `gml:id` and CityGML's native generic-attribute hook. Reverting them to the CityGML 3.0 dataType form would give both up, since a dataType carries no `gml:id` and cannot host generic attributes natively. This is the same family of issue as the ST2 gap above. A more realistic path is therefore to ADE the CityGML 3.0 Versioning module, extending its classes through the `_GenericApplicationPropertyOf...` hooks to restore identity and generic-attribute support where the schema needs them. This remains an educated guess and needs further investigation, since it depends on which `_CityObject` capabilities the schema actually relies on.

4. Implementing a CityJSON encoding that can then be benchmarked against the XML and SQL encodings. As discussed, this follows naturally after CityGML 3.0. It would require further evaluation or remapping of the versioning logic the Scenario ADE uses, as this

does not exist yet in CityJSON. The Scenario ADE would become a CityJSON Scenario Extension.

5. Build an application layer with a GUI for end users. This would handle querying, scenario merging, string and logic checking, conflict resolution, and the extraction of results back into engine inputs, none of which exist yet. It would also expose whether logic can easily resolve the limitations regarding generic attributes and ambiguity discussed in the residual schema gaps, and semantic rigity and domain agnostic sections. This thesis provides underlying functionality through its custom scripts to encode both XML and SQL. It also produced some initial SQL queries which interact with the Scenario ADE in 3DCityDB to show functionality. A application layer could re-use or build on top of this foundation.

7.3. Reflection

The Scenario ADE presented here is one valid solution, but most likely not the only one. The path to it was non-linear, and at several points a different but equally defensible choice could have been made. The design principles that guided and judged the schema were themselves drawn from scenario and urban planning literature and combined with a narrow scope of empirical evidence drawn from conducting simulations. A different interpretation of literature, or a case study from another domain, could have produced a different set of principles and a different schema. These alternatives are brought up throughout the thesis and in the discussion. The same holds true for the recommendations, which point at a gap rather than at a single fix. There is no single best fix for the Scenario ADE, trade-offs must be considered and logical context dependent decisions must derived.

What gives the result its confidence is not that it is the only answer but that every decision and direction taken was tied back to the principles and requirements and justified against them. Grounding the work in the energy domain is what made those requirements concrete, and it is also what bounds the claims. The schema reaches beyond energy by design, yet it has only been shown to hold within the domain that shaped it. Whether the same structure serves a flood, mobility or policy scenario as cleanly is the open question that the Scenario ADE Beta 6 now hands forward.

A Declaration of AI/LLM usage

Some AI tools were used during this thesis. Claude assisted in writing code for the simulation analysis and encoding pipeline. Gemini was used to improve the formulation of text, and at the end to reduce word count and improve the conciseness of certain sections. It also assisted in the brainstorming of conceptual ideas during the development and refinement stages of the schema.

All methods, results, analyses and conclusions are my own. I verified that the content is original and meets academic integrity standards, and that the AI tools did not generate falsified, fictional, or plagiarised content.

B Reproducibility self-assessment

All artifacts produced in this thesis are openly available in a public GitHub repository at https://github.com/fauer1-bit/Scenario-ADE-for-CityGML2.0. This repository holds the full Scenario ADE schema, its XML and SQL encodings, the SimStadt simulation run results, the SQL queries used to extract the results, the CSV tables those queries produced. All python scripts from simulation analysis, extracting simulation results into standardized tables, to encoding both CityGML files and enriching a 3DcityDB instance are also included.

Data availability. The simulation outputs, the encoded CityGML files, and the derived CSV tables are all present in the repository.

Code availability. All scripts used to run the queries, process the results, and generate the figures are in the repository. This includes the Python scripts that take the CSV result output through to the respective XML and SQL encodings. The Scenario ADE schema itself is provided in both its XML Schema and relational SQL form so that it can be inspected and reused directly.

Documentation. The repository README files describe the folder structure and the order in which the scripts are run. Reproducing the simulations themselves requires external software. SimStadt is used for the energy simulations, FME is used for the CityGML file filtering. PostgreSQL database with the 3D City Database is used for the relational storage and queries. These must all be installed seperately.

Reproducibility rating. I rate the reproducibility of this thesis as medium. Every result, encoding, and query is openly available and can be inspected and reused. All results presented are derived using open source software. Some processing steps such as filtering CityGML files was conducted in FME, which is proprietary.

B.1 Tools

The table below lists the tools used in this thesis with the versions where they are known. The exact Python library versions will be available in the `requirements.txt` and `pyproject.toml` files in the repository.

Software	Version
SimStadt	0.14.0
Python	3.13
3DCityDB	4.x (Importer/Exporter ≥ 4.4, ADE Manager plugin)
PostgreSQL / PostGIS	PostgreSQL ≥ 12
FME	2025.1.2
Enterprise Architect	13
ShapeChange	3.10

Table 34: Tools used in this thesis and their versions.

C Appendix

C.1 Xlink and Xpaths

Beta 5 relies on `newObjectMember` to apply attribute modifications by providing an entire replacement object. For example, changing a single U-value or window-to-wall ratio requires duplicating the entire building or wall geometry. This approach leads to massive data redundancy and excessive XML file bloat (Listing 19).

```xml
<scn:ScenarioMember>
  <scn:operation>modified</scn:operation>
  <scn:oldObjectMember xlink:href="#GML_Building_A1"/>
  <scn:newObjectMember>
    <bldg:Building gml:id="GML_Building_A1_v2">
      <bldg:lod2MultiSurface>...[Duplicating building geometry]...</bldg:lod2MultiSurface>
      <gen:doubleAttribute name="WWR">
        <gen:value>0.20</gen:value>
      </gen:doubleAttribute>
    </bldg:Building>
  </scn:newObjectMember>
</scn:ScenarioMember>
```

(1) Duplication of Geometry

Listing 19: Scenario ADE beta 5 duplication of building

A solution is inspired by the CityGML 3.0 Dynamizer module where a Xpath pointer is used to animated scalar data into timeseries data. In the case of the Scenario ADE, by pointing directly to the exact location of the attribute that needs changing, the original object geometry remains untouched, and the schema can apply a clean modification (Listing 20).

```xml
<scn:ScenarioMember>
  <scn:operation>modified</scn:operation>
  <scn:targetObject xlink:href="#GML_Building_A1"/>
  <scn:xpathPointer>//gen:doubleAttribute[@name='WWR']/gen:value</scn:xpathPointer>
  <scn:newValue>
    <scn:DoubleParameter>
      <scn:value>0.20</scn:value>
    </scn:DoubleParameter>
  </scn:newValue>
</scn:ScenarioMember>
```

(1) XPath Pointer

Listing 20: Scenario ADE beta 5 XPath example

The final Beta 6 design keeps the clean modification of the XPath option but replaces the `xpathPointer` string with a `parentFeature` xlink to the target feature and a named `attributeName`, carrying the new value through the typed `AttributeValue` union (Listing 21). The geometry stays untouched, and the change is anchored on the feature's `gml:id` rather than

on a path expression. The weakness that remains is that `attributeName` is itself an unconstrained string and that if the

```xml
<!-- sits inside the scn:VersionTransition -->
<scn:AttributeTransaction gml:id="id_at_wwr">
  <scn:type codeSpace=".../codelists/TransactionTypeValue.xml">replace</scn:type>
  <scn:parentFeature xlink:href="#GML_Building_A1"/>
  <scn:attributeType>cityGMLGenericAttribute</scn:attributeType>
  <scn:attributeName>WWR</scn:attributeName>
  <scn:value>
    <scn:AttributeValue>
      <scn:real>0.20</scn:real>
    </scn:AttributeValue>
  </scn:value>
</scn:AttributeTransaction>
```

Listing 21: Scenario ADE beta 6 AttributeTransaction example

Three mechanisms for changing a single attribute were available across the refinement. Table 35 shows each on the same change, a wall window-to-wall ratio set to 0.20, with the full encodings given above.

Option	Mechanism	Geometry	Targeting	Main weakness
1. Full replacement (Beta 5)	`newObjectMember` carries a whole replacement `Building`	duplicated	whole object	redundancy and file bloat
2. XPath pointer (Beta 5 probe)	`xpathPointer` string + `newValue`	untouched	exact attribute by path	`xpathPointer` is an unconstrained string. Only needed for replacing and delete existing attributes.
3. AttributeTransaction (Beta 6, chosen)	`parentFeature` xlink + `attributeName` + typed `value`	untouched	feature by `gml:id` plus named attribute	`attributeName` is an unconstrained string

Table 35: Three options for encoding an attribute-level change, shown on a wall window-to-wall ratio set to 0.20.

In the end, option 3 was adopted during the process into Beta 6. The `parentFeature` xlink anchors the change on the stable `gml:id` that the rest of the schema already rely on. The full precision of XPath can however also be useful in cases where a attribute needs to be replaced or deleted. In the four case study scenarios every change targets a feature with a stable `gml:id` and can be anchored through a `parentFeature` xlink with an `insert`, `delete`, or `replace` type. The ScenarioADE holds deltas and results on the scenario side. If something is modified it sits on the ScenarioADE and is applied only if it is realized, which is a task for the application layer. Cases where a feature has no stable `parentFeature` reference, such as address data, external references, or inline data types, which are affected by a scenario do not arise

in the four test scenarios. Where geometric changes are needed, removing the whole element through a `FeatureTransaction` is the appropriate solution instead of using a Xpath to target a specific point embedded in the position list (`gml:posList`) of any `GeometricPrimitive`. One limitation however still exists for referring to `citygmlGenericAttributes` which is a DataType. This and the one weakness it keeps from the unconstrained `attributeName`, is carried into the discussion (Section 6).

C.2 Window to Wall Ratio Outlier Detection

Building ID	Building Type	Built Year	Deviation (%)
ID_8eb33712-1727-4404-b96f-a3005904a81d	industry	2005	−64.6
BAG_0599100000752471	industry	2008	−61.5
ID_ad631dcc-539e-4e9e-a6be-cbc0e722ffbc	residential	2022	−53.5
ID_b8b627d8-e70f-4d49-a881-a1d943cb792d	residential	2022	−53.5
ID_dff176aa-e6e7-4a22-a379-5bc61aa0f8a8	residential	2022	−49.7

Table 36: WWR Outliers for Feijenoord (7P vs AI)

Building ID	Building Type	Built Year	Deviation (%)
ID_cdff71b6-d05d-47ba-89c8-9fe2db9b66ad	residential	2016	+465.3
ID_c0d2f88e-78d8-47e7-a1c7-0169b5f74bd2	residential	2016	+409.0
ID_24db115b-c59f-41f2-8071-670ec4d607d3	residential	1993	+370.8
ID_24f78bc9-c156-414d-816e-f35fcdc40ccc	residential	2016	+354.3
ID_5c2cbd20-b6c1-45db-9985-0b206cdaf000	residential	1993	+353.3

Table 37: Cooling Outliers for Feijenoord (7P vs AI)

Building ID	Building Type	Built Year	Deviation (%)
ID_2924a936-3a8f-4e30-8658-aefd5b2f32f1	production building	1998	+61.2
BAG_0599100000763580	production building	1997	−56.4
BAG_0599100000763583	production building	1997	−56.3
BAG_0599100000763585	production building	1997	−54.2
BAG_0599100000763618	production building	1997	+48.1
BAG_0599100000763623	production building	1997	+47.7

Table 38: Heating Outliers for Prinsenland (10P vs AI)

Building ID	Building Type	Built Year	Deviation (%)
ID_76110b6a-64a3-49b2-9038-1b3bfe7be611	residential	2006	+701.4
ID_3377ac94-5681-46d9-b536-c94e9fde2a36	residential	2006	+690.1
ID_87d3da38-4d09-41e7-bf95-d7c4015af698	residential	2012	+685.5
ID_c6f8e4be-54f4-4547-a4c8-a7ed84877e32	residential	2006	+648.8
ID_8076b199-cf65-4ef3-a076-2747e896f451	residential	1995	+597.8

Table 39: Cooling Outliers for Prinsenland (10P vs AI)

Building ID	Building Type	Built Year	Deviation (%)
BAG_0599100100011018	production building	2016	+103.5
BAG_0599100100002862	production building	1984	+81.1
ID_eaac1468-7fbe-4e58-b17c-9116d0b2143f	industry	1997	+55.7
ID_20f3048b-cf06-44ab-a91d-03437f32eba1	industry	1993	+52.9
ID_ea0c9dc0-af07-409d-9a0a-93387320b5bc	industry	1997	+49.8

Table 40: Heating Outliers for Feijenoord (20P vs AI)

Building ID	Building Type	Built Year	Deviation (%)
ID_cdff71b6-d05d-47ba-89c8-9fe2db9b66ad	residential	2016	+1505.3
ID_c0d2f88e-78d8-47e7-a1c7-0169b5f74bd2	residential	2016	+1315.7
ID_24db115b-c59f-41f2-8071-670ec4d607d3	residential	1993	+1240.6
ID_5c2cbd20-b6c1-45db-9985-0b206cdaf000	residential	1993	+1183.9
ID_5c39ebac-c7fd-4c5e-8524-5add167d6cae	residential	2014	+1142.0

Table 41: Cooling Outliers for Feijenoord (20P vs AI)

AI to 7 Feijenoord Heating and Cooling

Building ID	AI (MWh)	Std (MWh)	Diff %	Diff (MWh)	Usage	Year
ID_8eb33712-…81d	27.85	9.85	−64.6	−18.00	industry	2005
BAG_0599...2471	3.55	1.37	−61.5	−2.18	industry	2008
ID_ad631dcc-…fbc	5.43	2.53	−53.5	−2.91	residential	2022

Table 42: AI to 7% Feijenoord Heating Outliers (top 3)



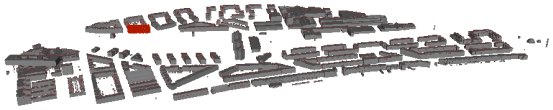


Figure C.1: Top heating outlier, AI to 7% (aerial photo and 3D model view).

Building ID	AI (MWh)	Std (MWh)	Diff %	Diff (MWh)	Usage	Year
ID_cdff71b6-…6ad	0.07	0.42	465.3	0.35	residential	2016
ID_c0d2f88e-…bd2	0.09	0.45	409.0	0.36	residential	2016
ID_24db115b-…7d3	0.10	0.45	370.8	0.36	residential	1993

Table 43: AI to 7% Feijenoord Cooling Outliers (top 3)

AI to 20 Feijenoord Heating and Cooling

Building ID	AI (MWh)	Std (MWh)	Diff %	Diff (MWh)	Usage	Year
BAG_0599...1018	0.12	0.23	103.5	0.12	production building	2016
BAG_0599...2862	0.98	1.77	81.1	0.79	production building	1984
ID_eaac1468-…43f	6.49	10.11	55.7	3.61	industry	

Table 44: AI to 20% Feijenoord Heating Outliers (top 3)

Building ID	AI (MWh)	Std (MWh)	Diff %	Diff (MWh)	Usage	Year
ID_cdff71b6-… 6ad	0.07	1.20	1505.3	1.13	residential	2016
ID_c0d2f88e-… bd2	0.09	1.26	1315.7	1.17	residential	2016
ID_24db115b-… 7d3	0.10	1.29	1240.6	1.19	residential	

Table 45: AI to 20% Feijenoord Cooling Outliers (top 3)

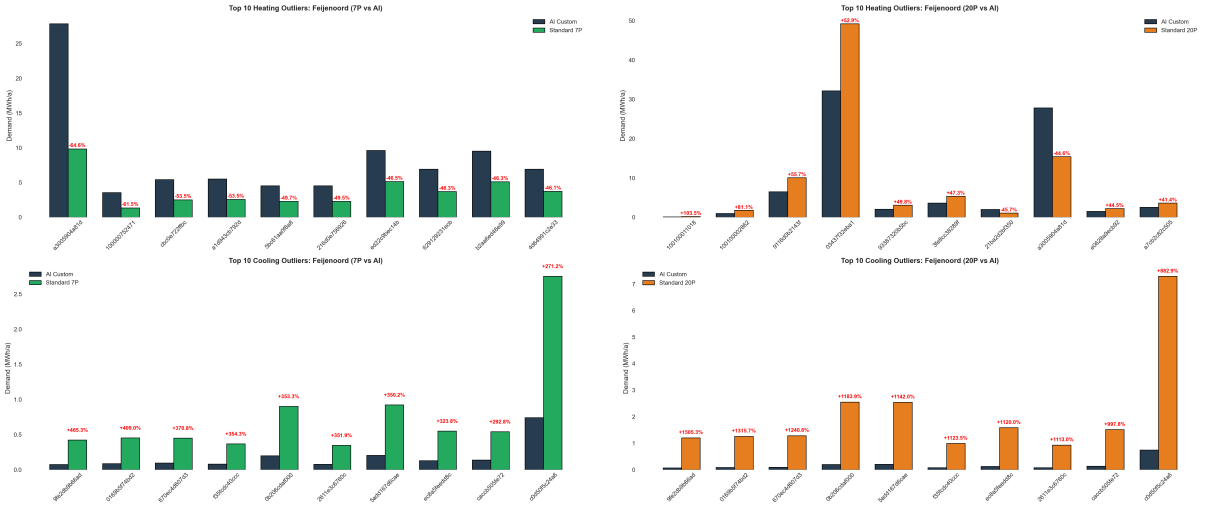


Figure C.2: Feijenoord top-10 heating and cooling outliers. Left: AI to 7 percent window to wall ratio. Right: AI to 20 percent window to wall ratio.

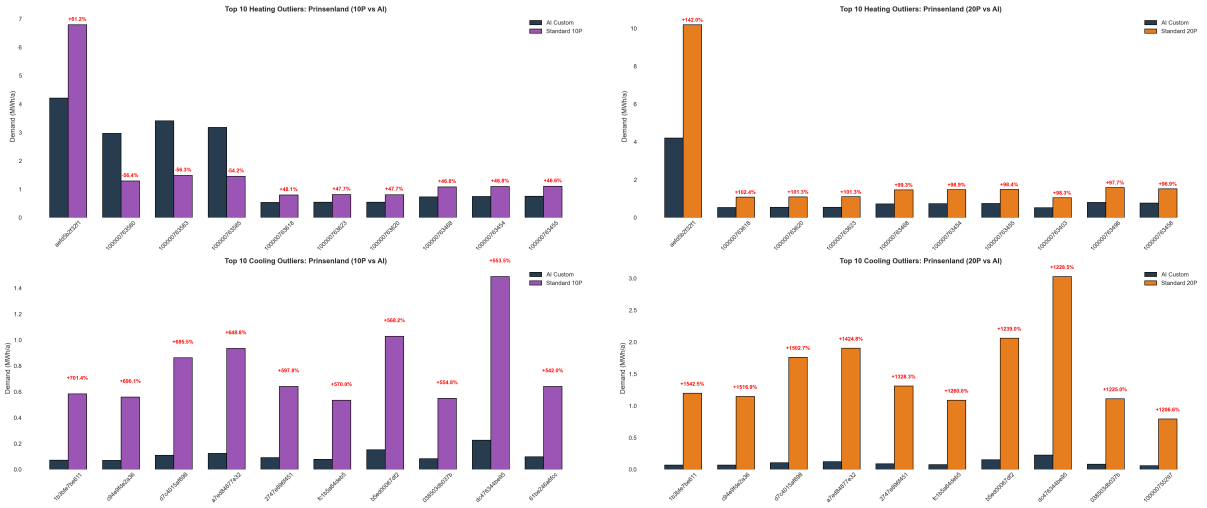


Figure C.3: Prinsenland top-10 heating and cooling outliers. Left: AI to 10 percent window to wall ratio. Right: AI to 20 percent window to wall ratio.

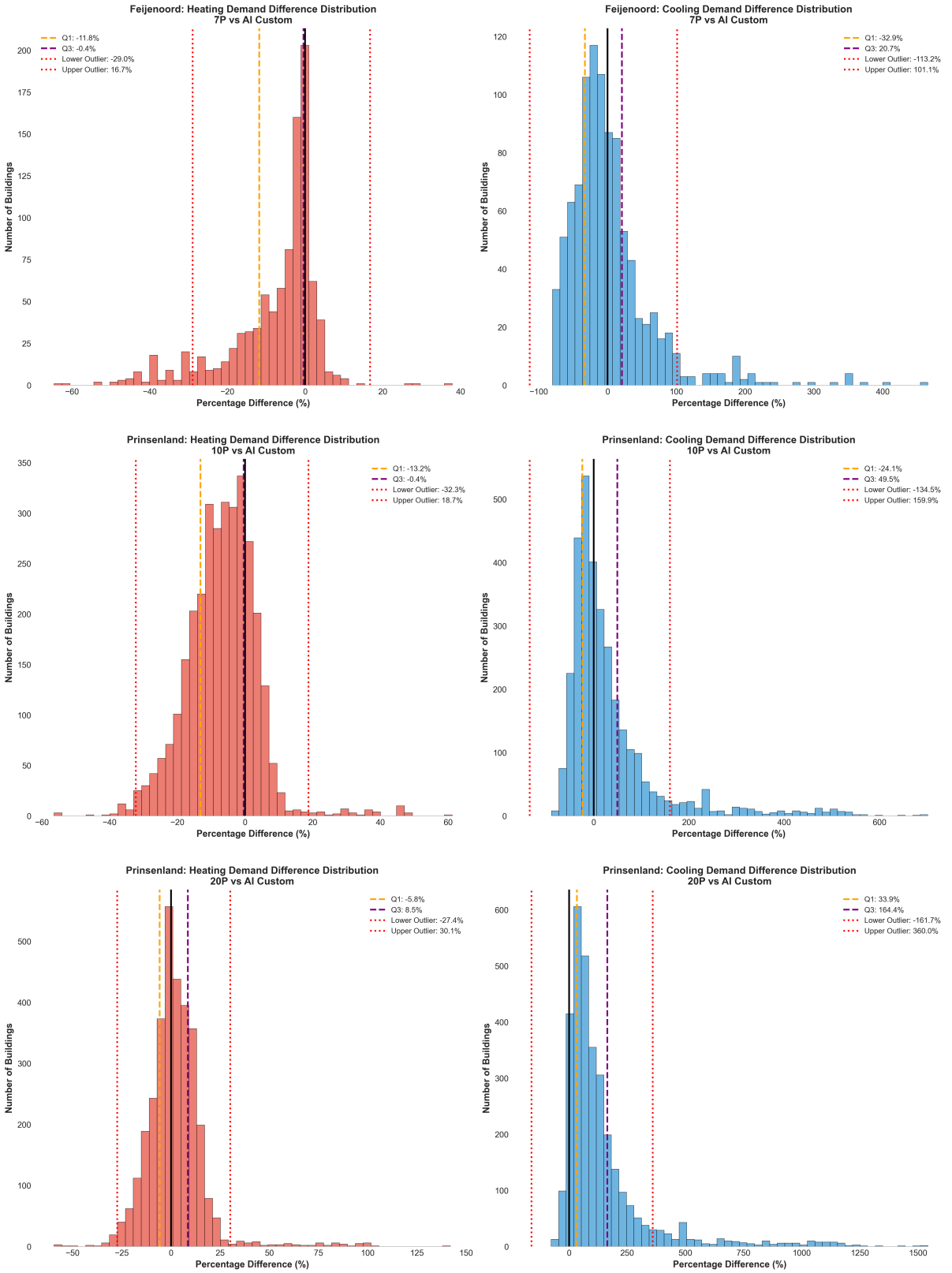


Figure C.4: Per-building demand difference distributions for the remaining WWR configurations (heating left, cooling right). Top: Feijenoord 7%. Middle: Prinsenland 10%. Bottom: Prinsenland 20%. The long cooling tail seen in Figure 5.5 recurs in every case.

D Appendix (SQL Queries)

```sql
-- Here is a list of all the queries ran, associated to they design principle, respectively.

-- Reusability of Scenario Settings and Simulation Engine (CF1)
SELECT
    scn_co.gmlid AS scenario,
    eng_co.gmlid AS engine,
    set_co.gmlid AS settings
FROM   scn_scenario_results sr
JOIN   cityobject scn_co ON scn_co.id = sr.scenario_id
JOIN   cityobject eng_co ON eng_co.id = sr.simulation_engine_id
JOIN   cityobject set_co ON set_co.id = sr.settings_id
ORDER  BY scenario;

-- Results are Traceable (CF2)
SELECT
    bldg.gmlid        AS building_gmlid,
    tx.attribute_name AS attr_name,
    tx.val_real       AS attr_value,
    tx.uom            AS uom,
    scn_co.gmlid      AS scenario_gmlid,
    eng_co.gmlid      AS engine_gmlid,
    set_co.gmlid      AS settings_gmlid
FROM   scn_transaction tx
JOIN   scn_transaction_to_cityobject t2c ON t2c.transaction_id = tx.id
JOIN   cityobject bldg   ON bldg.id   = t2c.cityobject_id
JOIN   scn_scenario_results sr ON sr.id = tx.scenario_results_id
JOIN   cityobject scn_co ON scn_co.id = sr.scenario_id
JOIN   cityobject eng_co ON eng_co.id = sr.simulation_engine_id
JOIN   cityobject set_co ON set_co.id = sr.settings_id
WHERE  tx.attribute_type = 'simpleAttribute'
AND  tx.val_real IS NOT NULL
AND  bldg.gmlid = 'BAG_0599100000618980'   -- for one building
ORDER  BY scenario_gmlid, attr_name;

-- Results are Distinguishable (CF3)
SELECT
    sr_co.gmlid AS scenario_results_gmlid,
    bldg.gmlid  AS building_gmlid,
    tx.attribute_name AS result_name,
    COUNT(*) AS n_rows
FROM scn_transaction tx
JOIN scn_transaction_to_cityobject t2c ON t2c.transaction_id = tx.id
JOIN cityobject bldg ON bldg.id = t2c.cityobject_id
JOIN scn_scenario_results sr ON sr.id = tx.scenario_results_id
JOIN cityobject sr_co ON sr_co.id = sr.id
WHERE tx.objectclass_id = 20130
  AND  tx.scenario_results_id IS NOT NULL
GROUP BY sr_co.gmlid, bldg.gmlid, tx.attribute_name
ORDER BY building_gmlid, result_name
LIMIT 12;

-- Finding the ancestors of a scenario (F2)
WITH RECURSIVE anchor AS (
    SELECT v.id AS version_id
    FROM   scn_scenario s
    JOIN   cityobject  s_co ON s_co.id = s.id
    JOIN   scn_version v   ON v.id    = s.version_id
    WHERE  s_co.id = '443'              -- id for PV adoption scenario.
),
ancestors AS (
    SELECT a.version_id, NULL::bigint AS via_transition_id, 0 AS depth
    FROM   anchor a
    UNION ALL
    SELECT vt.from_version_id, vt.id, anc.depth + 1
    FROM   ancestors anc
    JOIN   scn_version_transition vt ON vt.to_version_id = anc.version_id
    WHERE  vt.from_version_id IS NOT NULL
)
SELECT a.depth,
    v_co.gmlid                  AS version_gmlid,
    v.tag                       AS version_tag,
```

```
    vt_co.gmlid                  AS via_transition_gmlid,
    vt.type                      AS transition_type,
    vt.reason                    AS transition_reason
FROM   ancestors a
JOIN   scn_version v        ON v.id    = a.version_id
JOIN   cityobject  v_co     ON v_co.id = v.id
LEFT  JOIN scn_version_transition vt ON vt.id = a.via_transition_id
LEFT  JOIN cityobject  vt_co ON vt_co.id = a.via_transition_id
ORDER  BY a.depth;

-- Find all scenarios that branch from a given version (F2)
WITH RECURSIVE seed AS (
    SELECT v.id AS version_id
    FROM   scn_version v
    JOIN   cityobject  v_co ON v_co.id = v.id
    WHERE  v_co.id = '207'
),
descendants AS (
    SELECT s.version_id, NULL::bigint AS via_transition_id, 0 AS depth
    FROM   seed s
    UNION ALL
    SELECT vt.to_version_id, vt.id, d.depth + 1
    FROM   descendants d
    JOIN   scn_version_transition vt ON vt.from_version_id = d.version_id
)
SELECT d.depth,
    v_co.gmlid          AS version_gmlid,
    vt_co.gmlid         AS via_transition_gmlid,
    scn_co.gmlid        AS scenario_gmlid,
    scn.project_name    AS scenario_project
FROM   descendants d
JOIN   scn_version v        ON v.id    = d.version_id
JOIN   cityobject  v_co     ON v_co.id = v.id
LEFT  JOIN scn_version_transition vt    ON vt.id    = d.via_transition_id
LEFT  JOIN cityobject  vt_co            ON vt_co.id = d.via_transition_id
LEFT  JOIN scn_scenario scn             ON scn.version_id = v.id
LEFT  JOIN cityobject  scn_co           ON scn_co.id = scn.id
ORDER  BY d.depth, version_gmlid;

-- Context and strategy differentiable (C1)
-- Context Query
SELECT p.name,
       COALESCE(p.val_string, p.val_uri, p.val_real::text) AS value,
       p.scope
FROM   scn_scenario s
JOIN   cityobject co            ON co.id = s.id
JOIN   scn_scenario_results sr  ON sr.scenario_id = s.id
JOIN   scn_parameter p          ON p.settings_id = sr.settings_id
WHERE  co.gmlid = 'scn_refurbishment'
  AND  p.scope = 'input'
ORDER  BY p.name;

-- Strategy Query
SELECT vt.type            AS transition_type,
       vt.reason          AS transition_reason,
       tx.type            AS change,
       tx.attribute_name  AS attribute,
       tx.val_real        AS value,
       tx.uom,
       target.gmlid       AS target_feature
FROM   scn_scenario s
JOIN   cityobject co             ON co.id = s.id
JOIN   scn_version_transition vt ON vt.id = s.version_transition_id
JOIN   scn_transaction tx        ON tx.version_transition_id = vt.id
JOIN   scn_transaction_to_cityobject t2c ON t2c.transaction_id = tx.id
JOIN   cityobject target         ON target.id = t2c.cityobject_id
WHERE  co.gmlid = 'scn_refurbishment'
  AND  tx.attribute_name = 'meanUValue'
  LIMIT 3;


-- Understanding the Scenario adn their temporal horizons (C2)

SELECT co.gmlid       AS scenario,
       co.name        AS scenario_name,
       s.type         AS scenario_type,
       sco.valid_from,
       sco.valid_to,
```

```
150         co.description AS gmldescription_renamed_to_narrative
151 FROM    scn_scenario   s
152 JOIN    cityobject     co  ON co.id  = s.id
153 JOIN    scn_cityobject sco ON sco.id = s.id
154 ORDER  BY sco.valid_from, co.gmlid;
155
156 -- Inputs and Outputs are differentiable (C3)
157 SELECT
158     lib_co.gmlid                AS library_gmlid,
159     set_co.gmlid                AS settings_gmlid,
160     p.scope                     AS scope,
161     COUNT(*)                    AS n_parameters,
162     array_agg(p.name ORDER BY p.name) AS parameter_names
163 FROM       scn_library          lib
164 JOIN       cityobject           lib_co  ON lib_co.id = lib.id
165 JOIN       scn_simulation       settings ON settings.library_id = lib.id
166                                        AND settings.objectclass_id = 20220
167 JOIN       cityobject           set_co  ON set_co.id = settings.id
168 JOIN       scn_parameter        p       ON p.settings_id = settings.id
169 GROUP BY lib_co.gmlid, set_co.gmlid, p.scope
170 ORDER BY library_gmlid, settings_gmlid, scope;
171
172 -- Interventions reference existing features by id, geometry is not re-encoded (ST1)
173 SELECT
174     tx_co.gmlid   AS transaction_gmlid,
175     tx.type       AS change_type,
176     old_co.gmlid  AS old_feature,
177     new_co.gmlid  AS new_feature
178 FROM      scn_transaction tx
179 JOIN      cityobject  tx_co  ON tx_co.id  = tx.id
180 LEFT JOIN cityobject  old_co ON old_co.id = tx.old_feature_id
181 LEFT JOIN cityobject  new_co ON new_co.id = tx.new_feature_id
182 WHERE     tx.objectclass_id = 20120      -- FeatureTransaction
183 ORDER BY  change_type, transaction_gmlid
184 LIMIT 5;
185
186 -- ST2: each attribute change names its attribute and its exact target feature
187 SELECT
188     pf_co.gmlid       AS target_feature,
189     tx.attribute_name AS attribute,
190     tx.val_real       AS value,
191     tx.uom            AS uom
192 FROM      scn_transaction tx
193 JOIN      scn_transaction_to_cityobject t2c ON t2c.transaction_id = tx.id
194 JOIN      cityobject  pf_co ON pf_co.id = t2c.cityobject_id
195 WHERE     tx.objectclass_id      = 20130   -- AttributeTransaction
196   AND     tx.version_transition_id IS NOT NULL
197   AND     tx.scenario_results_id  IS NULL
198 ORDER BY  target_feature, attribute
199 LIMIT 5;
200
201
202 -- Comparing Cooling Demands
203
204  WITH results AS (
205       SELECT
206           target.gmlid        AS target_gmlid,
207           oc.classname        AS target_class,
208           tx.attribute_name   AS attr_name,
209           tx.val_real         AS attr_value,
210           tx.uom              AS uom,
211           scn_co.gmlid        AS scenario_gmlid
212       FROM       scn_transaction               tx
213       JOIN       scn_transaction_to_cityobject t2c    ON t2c.transaction_id = tx.id
214       JOIN       cityobject                    target ON target.id          = t2c.cityobject_id
215       JOIN       objectclass                   oc     ON oc.id              = target.objectclass_id
216       JOIN       scn_scenario_results          sr     ON sr.id              = tx.scenario_results_id
217       JOIN       scn_scenario                  scn    ON scn.id             = sr.scenario_id
218       JOIN       cityobject                    scn_co ON scn_co.id          = scn.id
219       WHERE  tx.objectclass_id = 20130
220     AND tx.attribute_name = 'yearlyCoolingDemand'
221   )
222   SELECT
223       a.target_class,
224       a.target_gmlid,
225       a.attr_name,
226       a.uom,
227       a.attr_value                    AS Refurbishment_Scenario_2050,
```

```
      b.attr_value                    AS Future_Climate_Scenario_2050,
      (b.attr_value - a.attr_value)   AS delta,
      CASE WHEN a.attr_value <> 0
           THEN 100.0 * (b.attr_value - a.attr_value) / a.attr_value
           ELSE NULL
      END                             AS pct_change
FROM       results a
INNER JOIN results b
        ON  b.target_gmlid = a.target_gmlid
       AND  b.attr_name    = a.attr_name
WHERE  a.scenario_gmlid = 'scn_refurbishment'
  AND  b.scenario_gmlid = 'scn_future_climate'
ORDER  BY a.target_class, a.target_gmlid, a.attr_name
LIMIT 5;
```

Listing 22: Scenario Lineage

Bibliography

Abbasabadi, N., & Ashayeri, M. (2019). Urban Energy Use Modeling Methods and Tools: A Review and an Outlook. *Building and Environment*, *161*, 106270. https://doi.org/10.1016/j.buildenv.2019.106270

Abou Jaoude, G., Mumm, O., & Carlow, V. M. (2022). An Overview of Scenario Approaches: A Guide for Urban Design and Planning. *Journal of Planning Literature*, *37*(3), 467–487. https://doi.org/10.1177/08854122221083546

Agugiaro, G., & Padsala, R. (2025). A Proposal to Update and Enhance the CityGML Energy Application Domain Extension. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, 1–8. https://doi.org/10.5194/isprs-annals-X-4-W6-2025-1-2025

Agugiaro, G., Benner, J., Cipriano, P., & Nouvel, R. (2018). The Energy Application Domain Extension for CityGML: Enhancing Interoperability for Urban Energy Simulations. *Open Geospatial Data, Softw. Stand.*, *3*(1), 2. https://doi.org/10.1186/s40965-018-0042-y

Bachert, C., León-Sánchez, C., Kutzner, T., & Agugiaro, G. (2024). Mapping the CityGML Energy ADE to CityGML 3.0 Using a Model-Driven Approach. *ISPRS International Journal of Geo-Information*, *13*(4), 121. https://doi.org/10.3390/ijgi13040121

Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., & others. (2013). Prov-dm: The prov data model. *W3c Recommendation*, *14*, 15–16. https://www.w3.org/TR/prov-dm/

Bessembinder, J., Bintanja, R., van Dorland, R., Homan, C., Overbeek, B., Selten, F., & Siegmund, P. (2023). *KNMI'23 Klimaatscenario's voor Nederland Gebruikersrapport* (technical report Nos. 23–3.). https://cdn.knmi.nl/system/about_blocks/files/000/000/001/original/KNMI23_klimaatscenarios_gebruikersrapport_23-03.pdf?1760364457

Betz, M., Riegel, A., & Coors, V. (2025). Storing Quality Validation Results in CityGML with the QualityADE. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, 31–37. https://doi.org/10.5194/isprs-archives-XLVIII-4-W15-2025-31-2025

Biljecki, F., Kumar, K., & Nagel, C. (2018). CityGML Application Domain Extension (ADE): Overview of Developments. *Open Geospatial Data, Softw. Stand.*, *3*(1), 13. https://doi.org/10.1186/s40965-018-0055-6

Bradfield, R., Wright, G., Burt, G., Cairns, G., & Van Der Heijden, K. (2005). The Origins and Evolution of Scenario Techniques in Long Range Business Planning. *Futures*, *37*(8), 795–812. https://doi.org/10.1016/j.futures.2005.01.003

Börjeson, L., Höjer, M., Dreborg, K.-H., Ekvall, T., & Finnveden, G. (2006). Scenario types and techniques: Towards a user's guide. *Futures*, *38*(7), 723–739. https://doi.org/https://doi.org/10.1016/j.futures.2005.12.002

Chaturvedi, K., Smyth, C. S., Gesquière, G., Kutzner, T., & Kolbe, T. H. (2017). Managing Versions and History Within Semantic 3D City Models for the Next Generation of CityGML. In A. Abdul-Rahman (Ed.), *Advances in 3D Geoinformation: Advances in 3D Geoinformation* (pp. 191–206). Springer International Publishing. https://doi.org/10.1007/978-3-319-25691-7_11

Dahlström, L., Broström, T., & Widén, J. (2022). Advancing Urban Building Energy Modelling through New Model Components and Applications: A Review. *Energy and Buildings*, *266*, 112099. https://doi.org/10.1016/j.enbuild.2022.112099

Dang, M., Van Den Dobbelsteen, A., & Voskuilen, P. (2024). A Parametric Modelling Approach for Energy Retrofitting Heritage Buildings: The Case of Amsterdam City Centre. *Energies*, *17*(5), 994. https://doi.org/10.3390/en17050994

Dietze, L., Nonn, U., & Zipf, A. (2007). Metadata for 3D city models analysis of the applicability of the ISO 19115 standard and possibilities for further amendments. *Proceedings of the 10th AGILE International Conference on Geographic Information Science*, 1–9. https://homes.plan.aau.dk/enc/AGILE2007/PDF/87_PDF.pdf

Ebrahimigharehbaghi, S., Qian, Q. K., Meijer, F. M., & Visscher, H. J. (2019). Unravelling Dutch Homeowners' Behaviour towards Energy Efficiency Renovations: What Drives and Hinders Their Decision-Making?. *Energy Policy*, *129*, 546–561. https://doi.org/10.1016/j.enpol.2019.02.046

Ferrando, M., & Causone, F. (2019). An Overview Of Urban Building Energy Modelling (UBEM) Tools. *Building Simulation 2019*, 3452–3459. https://doi.org/10.26868/25222708.2019.210632

FIEC. (2022,). *Netherlands*. https://fiec-statistical-report.eu/2022/netherlands

Frayssinet, L., Merlier, L., Kuznik, F., Hubert, J.-L., Milliez, M., & Roux, J.-J. (2018). Modeling the Heating and Cooling Energy Demand of Urban Buildings at City Scale. *Renewable and Sustainable Energy Reviews*, *81*, 2318–2327. https://doi.org/10.1016/j.rser.2017.06.040

Gao, W., Agugiaro, G., & León-Sánchez, C. (2025). Data-Driven Energy Simulations To Evaluate Positive Energy District Potential In Rotterdam. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, 25–32. https://doi.org/10.5194/isprs-annals-X-4-W7-2025-25-2025

Godet, M. (1986). Introduction to la prospective: Seven key ideas and one scenario method. *Futures*, *18*(2), 134–157. https://doi.org/https://doi.org/10.1016/0016-3287(86)90094-7

Government of the Netherlands. (2019). *National Climate Agreement - The Netherlands* [Technical report]. https://english.rvo.nl/sites/default/files/2020/07/National%20Climate%20Agreement%20The%20Netherlands%20-%20English.pdf

Gröger, G., & Plümer, L. (2012). CityGML – Interoperable Semantic 3D City Models. *ISPRS Journal of Photogrammetry and Remote Sensing*, *71*, 12–33. https://doi.org/10.1016/j.isprsjprs.2012.04.004

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research1. *MIS Quarterly*, *28*(1), 75–106. https://doi.org/10.2307/25148625

Idris, N. H., Azri, S., & Ujang, U. (2024). Reviewing the Versioning in 3D City Model to Track Building History and Progress. *IOP Conf. Ser.: Earth Environ. Sci.*, *1412*(1), 12035. https://doi.org/10.1088/1755-1315/1412/1/012035

International Energy Agency. (2025). *World Energy Outlook 2025* [Technical report]. https://www.iea.org/reports/world-energy-outlook-2025

ISO. (2014). *ISO 19115-1:2014: Geographic information — Metadata — Part 1: Fundamentals* [Standard]. https://www.iso.org/standard/53798.html

Jiang, L., Yue, P., Kuhn, W., Zhang, C., Yu, C., & Guo, X. (2018). Advancing Interoperability of Geospatial Data Provenance on the Web: Gap Analysis and Strategies. *Computers & Geosciences*, *117*, 21–31. https://doi.org/10.1016/j.cageo.2018.05.001

Khakee, A. (1991). Scenario Construction for Urban Planning. *Omega*, *19*(5), 459–469. https://doi.org/10.1016/0305-0483(91)90062-X

Kong, D., Cheshmehzangi, A., Zhang, Z., Ardakani, S. P., & Gu, T. (2023). Urban Building Energy Modeling (UBEM): A Systematic Review of Challenges and Opportunities. *Energy Efficiency*, *16*(6), 69. https://doi.org/10.1007/s12053-023-10147-z

Labetski, A., Kumar, K., Ledoux, H., & Stoter, J. (2018). A Metadata ADE for CityGML. *Open Geospatial Data, Software and Standards*, *3*(1), 16. https://doi.org/10.1186/s40965-018-0057-4

Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A., & Vitalis, S. (2019). CityJSON: A Compact and Easy-to-Use Encoding of the CityGML Data Model. *Open Geospatial Data, Softw. Stand.*, *4*(1), 4. https://doi.org/10.1186/s40965-019-0064-0

León-Sánchez, C., Agugiaro, G., & Stoter, J. (2025). Lessons Learnt from the Integration of Open Data and Semantic 3D City Models for Urban Building Energy Modelling in the Netherlands. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, 89–96. https://doi.org/10.5194/isprs-annals-X-4-W7-2025-89-2025

Liping Di, Yuanzheng Shao, & Lingjun Kang. (2013). Implementation of Geospatial Data Provenance in a Web Service Workflow Environment With ISO 19115 and ISO 19115-2 Lineage Model. *IEEE Transactions on Geoscience and Remote Sensing*, *51*(11), 5082–5089. https://doi.org/10.1109/TGRS.2013.2248740

Luo, S., Ding, L., & Fan, H. (2025). A CityGML ADE for Flood Simulations. *Geo-Spatial Information Science*, 1–15. https://doi.org/10.1080/10095020.2025.2547952

Moriarty, J. P. (2012). Theorising Scenario Analysis to Improve Future Perspective Planning in Tourism. *Journal of Sustainable Tourism*, *20*(6), 779–800. https://doi.org/10.1080/09669582.2012.673619

Nouvel, R., Brassel, K.-H., Bruse, M., Duminil, E., Coors, V., Eicker, U., & Robinson, D. (2015). SimStadt, a new workflow-driven urban energy simulation platform for CityGML city models. *Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale*, 889–894. https://infoscience.epfl.ch/server/api/core/bitstreams/e06c9c38-35f3-4bcc-9a3c-154d1b430776/content

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Powałka, L., Poon, C., Xia, Y., Meines, S., Yan, L., Cai, Y., Stavropoulou, G., Dukai, B., & Ledoux, H. (2024). Cjdb: A Simple, Fast, and Lean Database Solution for the CityGML Data Model. In T. H. Kolbe, A. Donaubauer, & C. Beil (Eds.), *Recent Advances in 3D Geoinformation Science: Recent Advances in 3D Geoinformation Science*. https://doi.org/10.1007/978-3-031-43699-4_47

Rahmawati, A. (2025). *Scenario-based energy simulation: Modelling tree planting strategy to reduce heating and cooling demand under 2050 climate conditions* [Master's thesis]. https://repository.tudelft.nl/record/uuid:bf97005e-bd6c-4d76-b5e8-91a94c40d99f

Salvalai, G., Zhu, Y., & Maria Sesana, M. (2024). From Building Energy Modeling to Urban Building Energy Modeling: A Review of Recent Research Trend and Simulation Tools. *Energy and Buildings*, *319*, 114500. https://doi.org/10.1016/j.enbuild.2024.114500

Spaniol, M. J., & Rowland, N. J. (2019). Defining Scenario. *Futures & Foresight Science*, *1*(1), e3. https://doi.org/10.1002/ffo2.3

Tapinos, E. (2012). Perceived Environmental Uncertainty in Scenario Planning. *Futures*, *44*(4), 338–345. https://doi.org/10.1016/j.futures.2011.11.002

UN-Habitat. (2022). *World cities report 2022 : envisaging the future of cities*. xxxi, 387 p. http://digitallibrary.un.org/record/3984713

Van Den Brink, L., Stoter, J., & Zlatanova, S. (2013). \textsc{UML} -Based Approach to Developing a \textsc{CityGML} Application Domain Extension. *Transactions in GIS*, *17*(6), 920–942. https://doi.org/10.1111/tgis.12026

Varum, C. A., & Melo, C. (2010). Directions in Scenario Planning Literature – A Review of the Past Decades. *Futures*, *42*(4), 355–369. https://doi.org/10.1016/j.futures.2009.11.021

Vitalis, S., Labetski, A., Arroyo Ohori, K., Ledoux, H., & Stoter, J. (2019). A DATA STRUCTURE TO INCORPORATE VERSIONING IN 3D CITY MODELS. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, 123–130. https://doi.org/10.5194/isprs-annals-IV-4-W8-123-2019

Wack, P. (1985). Scenarios: uncharted waters ahead. *Harvard Business Review*, *63*(5), 72–89. https://hbr.org/1985/09/scenarios-uncharted-waters-ahead

Wang, C., Ferrando, M., Causone, F., Jin, X., Zhou, X., & Shi, X. (2022). Data Acquisition for Urban Building Energy Modeling: A Review. *Building and Environment*, *217*, 109056. https://doi.org/10.1016/j.buildenv.2022.109056

Wieland, M., Pittore, M., Wieland, M., & Pittore, M. (2017). A Spatio-Temporal Building Exposure Database and Information Life-Cycle Management Solution. *ISPRS International Journal of Geo-Information*, *6*(4). https://doi.org/10.3390/ijgi6040114

Yao, Z., Nagel, C., Kendir, M., Willenborg, B., & Kolbe, T. H. (2025). The New 3D City Database 5.0 - Advancing 3D City Data Management Based on CityGML 3.0. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 241–248. https://doi.org/10.5194/isprs-annals-X-4-W6-2025-241-2025

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., & Kolbe, T. H. (2018). 3DCityDB - a 3D Geodatabase Solution for the Management, Analysis, and Visualization of Semantic 3D City Models Based on CityGML. *Open Geospatial Data, Softw. Stand.*, *3*(1), 5. https://doi.org/10.1186/s40965-018-0046-7

Zvarikova, K., Horák, J., & Downs, S. (2022). Digital Twin Algorithms, Smart City Technologies, and 3D Spatio-Temporal Simulations in Virtual Urban Environments. *Geopolitics, History, And International Relations*, *14*(1), 139–154. https://www.ceeol.com/search/article-detail?id=1055695

This document was typeset using Typst, and uses the template freely available at https://github.com/tudelft3d/msc_geomatics_thesis_typst.