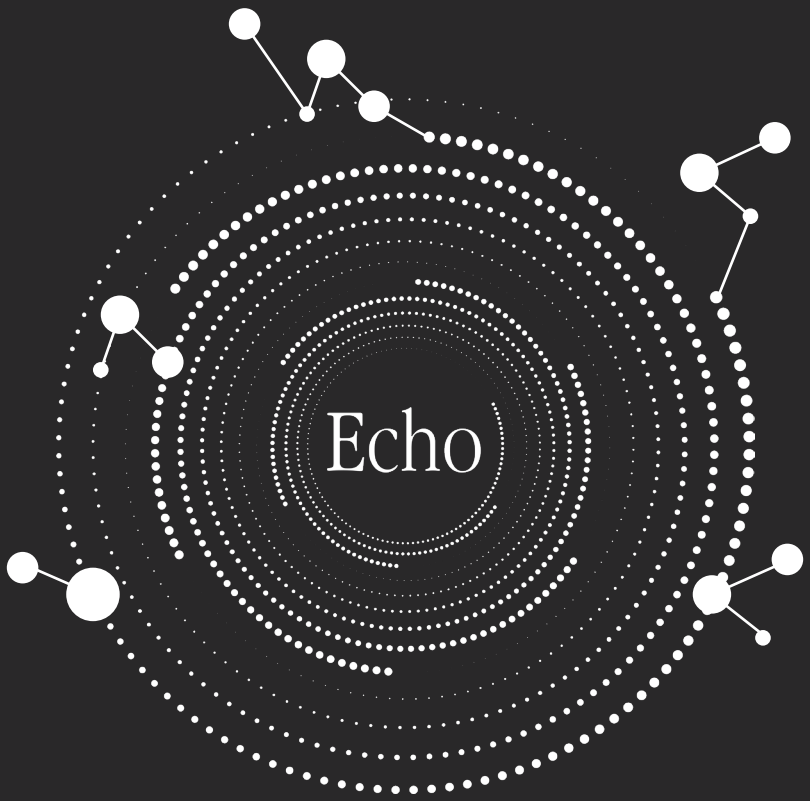
Building Technology Graduation Studio | June 2025
Symeon Maniatis



Real-Time Evaluation Of The Life-Cycle Performance And Material Usage Of Modular Design

A computational tool leveraging Graph Neural Networks to assist designers and stakeholders in early-stage design.

TU Delft
sustainer .home

Delft University of Technology
MSc Architecture, Urbanism & Building Sciences

Student:
Symeon Maniatis | 5916046

Mentors :
Dr. Michela Turrin | Design Informatics
Dr. Olga Ioannou | Architecture Engineering & Technology

Delegate of the Board of Examiners:
Henriette Bier

November 2024 - June 2025

Acknowledgement

I wish to express my sincere gratitude to my mentors, Dr. Michela Turrin and Dr. Olga Ioannou, for their unwavering support and invaluable guidance throughout this project. Their expertise, insightful feedback, and continuous encouragement were instrumental in addressing the challenges associated with researching the convergence of circular strategies and computational methods.

Dr. Turrin played an essential role in shaping a coherent narrative for the research and the development of our computational tool, deepening my understanding of its potential impact on the built environment industry. Meanwhile, Dr. Ioannou consistently pinpointed areas for improvement and offered constructive alternatives, transforming each setback into an opportunity for growth.

I also extend my heartfelt appreciation to Sustainer, whose innovative toolkit was fundamental to this research. In particular, I am grateful to Anastasia Florou, who served as my advisor and provided dedicated guidance and consultation throughout the project's timeline.

Thank you all for fostering such a collaborative and inspiring learning environment.

Abstract

Decision-making in early-stage design often lacks robust methods for evaluating circularity, resulting in outcomes that may not fully realize their potential for efficiency. This research presents the development of a computational tool or "Intelligent Design Assistant" that employs Graph Neural Networks (GNNs) to deliver real-time assessments of life-cycle performance and material usage for modular designs. By utilizing user-defined, simplified early-stage representations, the tool provides actionable insights into both design and environmental performance.

A central point of this approach is the adoption of a graph-based framework where each building module is represented as a node, and its interactions with neighboring modules are captured through connecting edges. This framework not only reflects the intrinsic properties of each module, but it also dynamically evaluates how a module's characteristics evolve based on its spatial and functional relationships. Although the study focuses on laminated veneer lumber (LVL)—selected for its extensive environmental data—the scalable machine learning model is designed to be applicable to a wide range of construction methods and materials.

Through experimental validation, the integration of GNNs has been shown to enhance early design decision-making by providing real-time feedback. The model achieves an accuracy of approximately 85% -90% under conditions similar to the training data. This capability enables designers, clients, and other stakeholders to engage in informed discussions about design modifications and circularity measures well before detailed construction planning begins, thereby promoting more sustainable and circular design practices across the industry.

Keywords: real-time assesment, life-cycle performance, material usage, laminated veneer lumber, early-stage design, modular, graph neural network, machine learning

Contents

Work Package 01| Literature Review & Requirements Gatherin

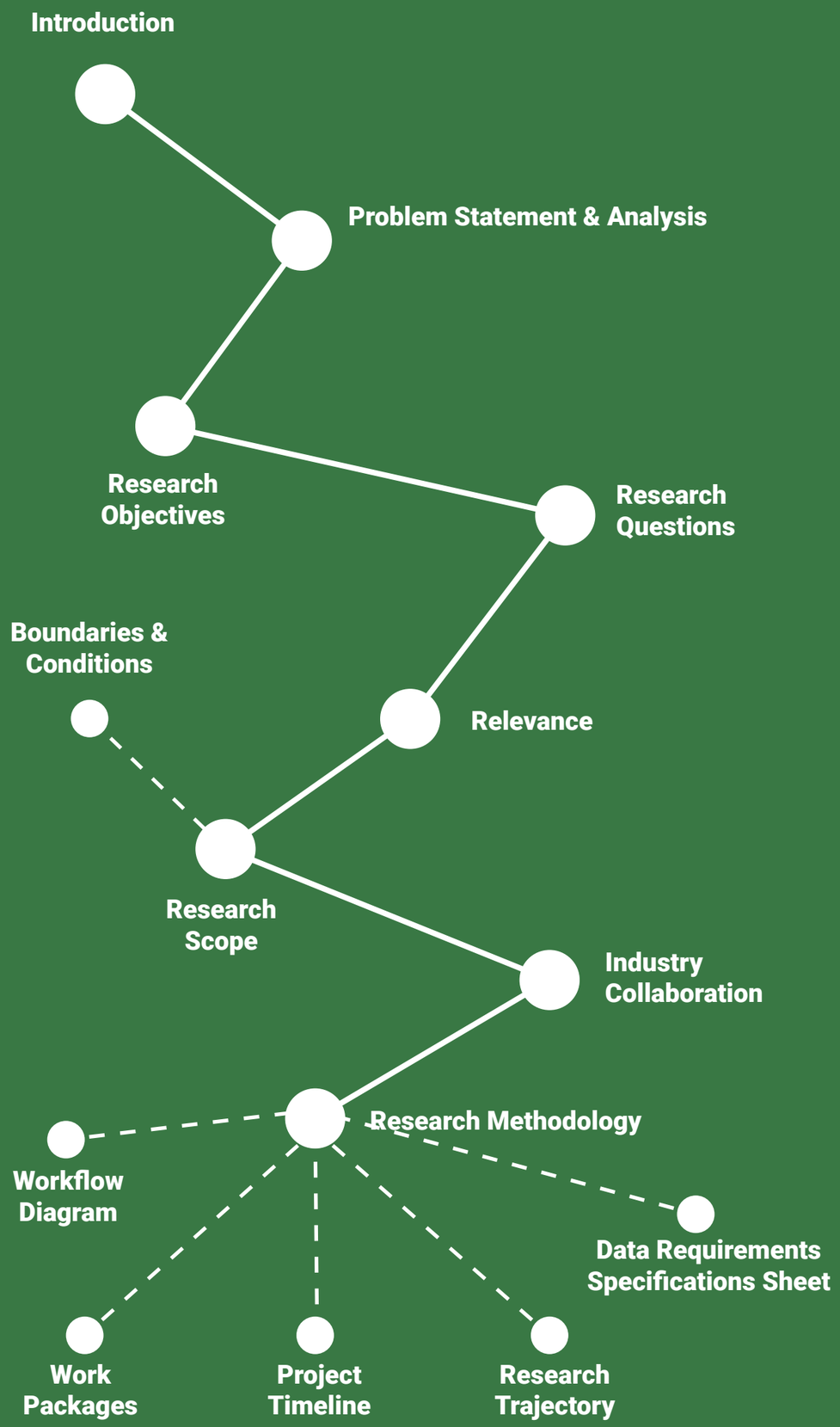
A. Introduction

Brief Summary

The introduction highlights the urgent need to embed rigorous life-cycle thinking into the earliest stages of building design. It begins by pointing to global sustainability challenges such as climate change, resource depletion, pollution and biodiversity loss and notes international policy responses including the Paris Agreement and the European Union's circular-economy plan. The construction sector is shown to play a central role, consuming a majority of extracted materials, using a large share of final energy and generating a significant portion of greenhouse gas emissions in Europe. Yet current practice treats detailed life-cycle assessments and material analyses as late-stage compliance exercises, making substantive design iteration costly and infrequent.

Advances in computational modeling, especially graph neural networks, offer the potential to provide designers with real-time feedback on complex relational data in building assemblies. However, the opacity of many machine learning models undermines user trust and practical adoption unless explainability mechanisms are integrated. At the same time, emerging engineered wood products such as laminated veneer lumber combine strong performance with circular-economy objectives but remain difficult to assess in early design workflows.

By bringing together policy objectives, computational insights and material innovations, the introduction argues for a new class of design tool that can deliver transparent, data-driven guidance from the very first sketches. Such an approach would transform sustainability from an afterthought into a core driver of architectural creation.



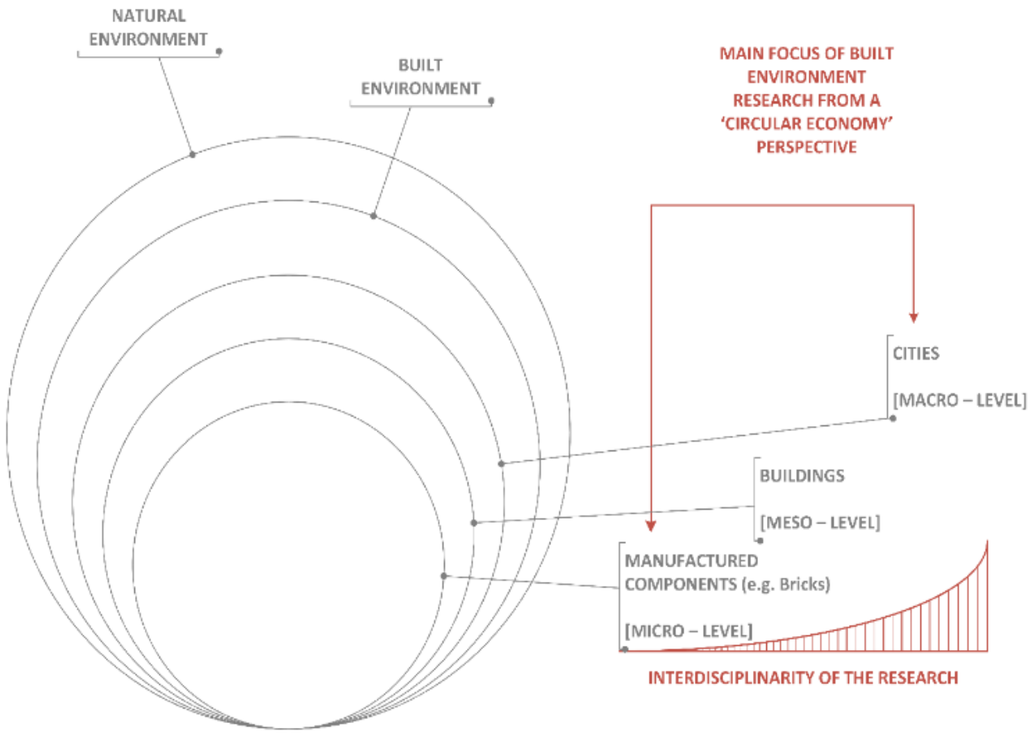


Figure 01: A research framework (Figure 2, p. 14), by F. Pomponi & A. Moncaster, 2017.

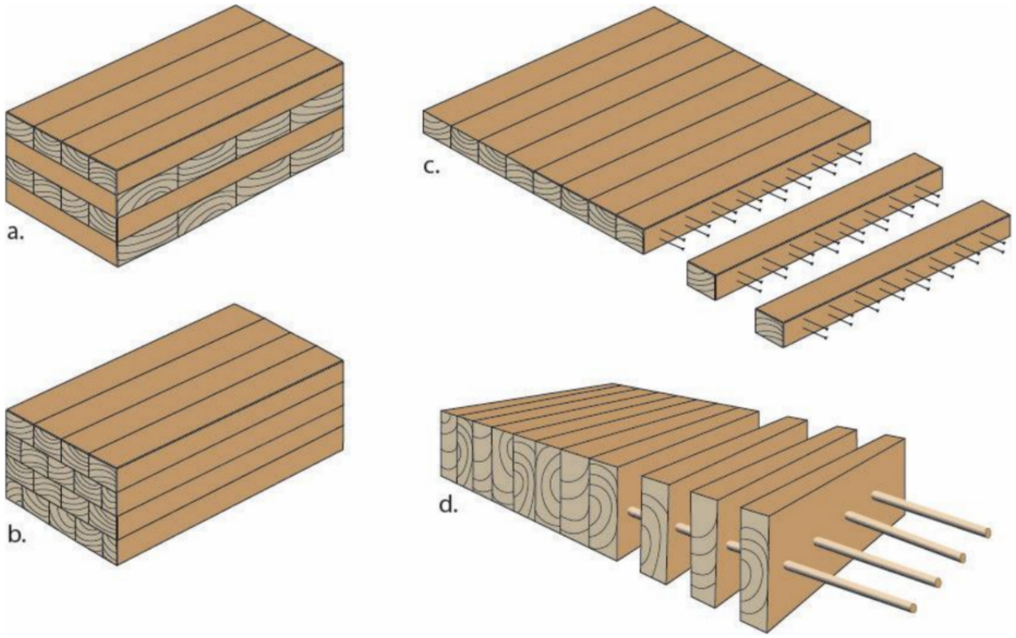


Figure 02: Wood Composites (Figure 2, p. 4), by Abed, Rayburg, Rodwell, & Neave, 2022

A. Introduction

A.1 Introduction

The segments below are taken from official project meetings during the research's development:

1.Discussion with ABT:

"Who ultimately owns circularity? **When we sit down with the other engineers, how do we weave circular principles into the design process?"**

"Right now we don't have a concrete method for embedding circularity at project kick-off. Designing with emerging materials such as cross-laminated timber (CLT) is still new to the market. Reuse raises a similar issue: it's hard to talk about reclaiming timber when most buildings that could supply it weren't built with wood in the first place. **Our answer, for now, is to bring circularity onto the agenda as early as possible—by engaging stakeholders and by setting clear, ambitious project goals."**

2.Discussion with Finch Buildings:

"**Would a tool that fine-tunes structural design and manufacturing,** so you can **cut carbon** and material use while boosting circular performance, be useful to your projects?"

"You're asking the wrong question. Naturally we want stronger environmental results. But manufacturers have refined their own methods for years; overhauling them would take immense effort for unclear benefit. **The real issue is that early architectural proposals arrive without solid data on metrics like these."**

Academic research is moving quickly, but industry often falls behind for several reasons. Manufacturing and construction, the biggest contributors to embodied carbon, can only be assessed rigorously once detailed data on materials and processes are in hand. Early in a project, when foundational design choices are still fluid, designers lack reliable metrics to gauge carbon performance. By the time accurate figures finally emerge, the design is already locked in, and making meaningful changes becomes prohibitively costly and complex.

The circular economy has emerged as a comprehensive framework designed to eliminate waste and pollution, circulate materials at their highest value, and regenerate natural systems (Kirchherr, Reike, & Hekkert, 2017). Yet, analyses of the EU's 2015 Circular Economy Action Plan reveal a persistent gap between lofty policy ambitions and on the ground implementation (Calisto Friant, Vermeulen, & Salomone, 2021).

Within this systems level context, the construction sector presents both a significant challenge and opportunity. It consumes over half of all extracted materials, accounts for 42% of final energy use, and generates roughly 35% of greenhouse gas emissions in Europe (Gervasio Dos Santos & Dimova, 2018). Despite these high stakes, circularity in architectural practice frequently remains aspirational and articulated in broad conceptual goals yet lacking the quantitative rigor needed to inform fundamental design choices

Part of this disconnect stems from current workflows, where rigorous life cycle assessments (LCAs) and material usage analyses, often based on Environmental Product Declarations are conducted only after structural systems, material choices, and manufacturing processes are effectively locked in. Consequently, LCAs become retrospective compliance exercises, making substantive iteration both costly and time consuming (Pomponi & Moncaster, 2017).

A. Introduction
A.2 Problem Statement And Analysis

Despite ambitious circular economy targets that prioritize material reduction and component reuse, very few design stage tools embed life cycle thinking at the moment when it matters most, during concept development. Consequently, circularity remains an aspirational sketch, while rigorous quantitative assessments such as Environmental Product Declarations (EPDs) or full Life Cycle Assessments (LCAs) are postponed until structural systems, materials, and manufacturing routes have already been fixed. By then, redesigns are costly, and LCAs are reduced to compliance checks instead of serving as drivers of innovation (Pomponi & Moncaster, 2017; Gervasio & Dimova, 2018).

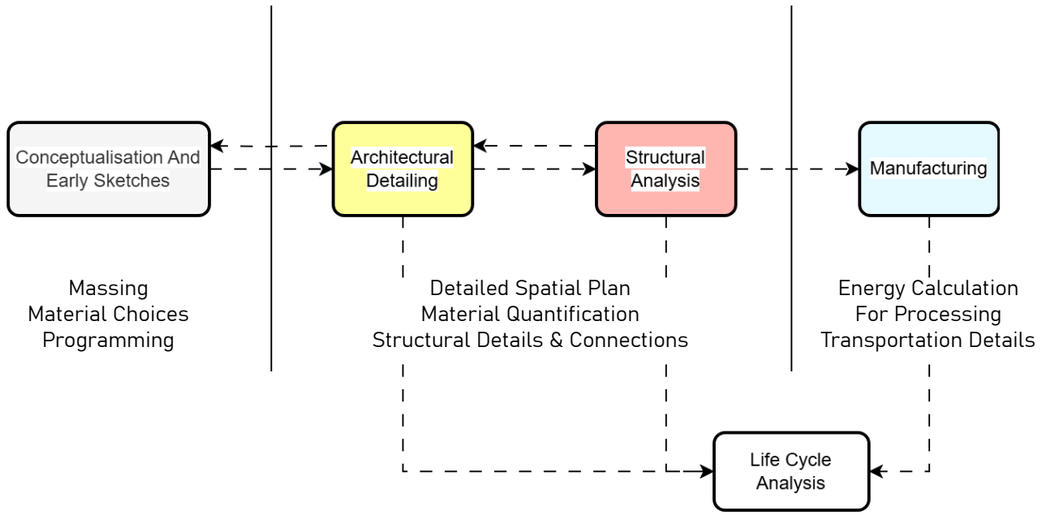


Figure 03: Lack of information for accurate LCA calculations in early stage design. Source: Author

This research focuses on the primary structural material for modular timber construction and requires extensive datasets and example configurations, complete with manufacturing details. The industrial collaborator, Sustainer, provided the means to generate hundreds of such modular design examples; its chosen primary structural material is Laminated Veneer Lumber (LVL). LVL is selected as the case material due to its high strength-to-weight ratio, dimensional stability, large panel format, compatibility with off-site manufacturing, and support for circular-economy objectives. Specialized expertise in LVL design and engineering is also contributed by Sustainer.

This disconnect between circular ambitions and early stage practice creates three inter related challenges:

1. Information insufficiency. Early stage models omit critical variables like embodied carbon, recyclability and end of life scenarios that force architects and engineers to make decisions under deep uncertainty and with no link to spatial or schematic representations (Pomponi & Moncaster, 2017).

2.Delayed feedback. Life cycle insights arrive too late to influence core design choices, stifling innovation and relegating sustainability to a perfunctory audit (Gervasio & Dimova, 2018).

3.Data fragmentation. Disparate silos, EPD repositories, structural analysis software, manufacturer specifications block the seamless data flow needed for iterative, real time evaluation (Kirchherr et al., 2018).

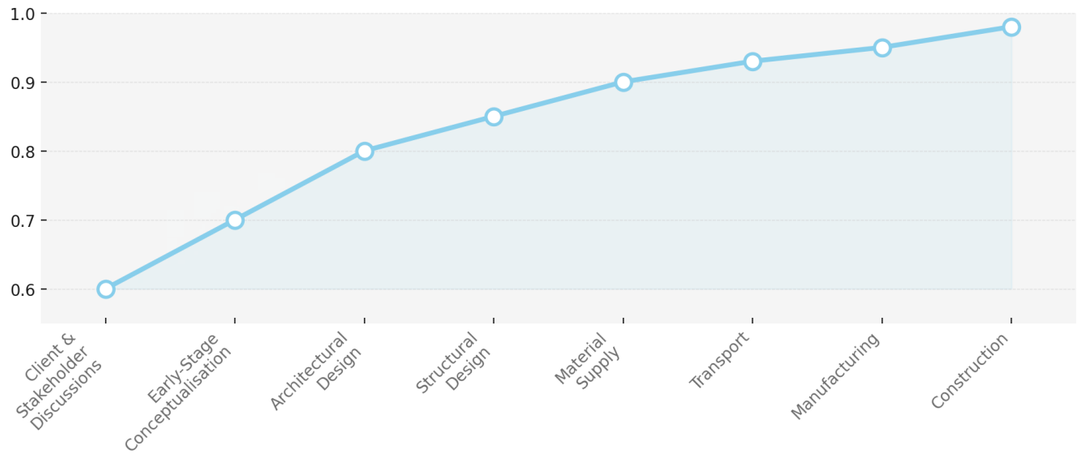


Figure 04: The increase of difficulty to improve LCA performance over-time in a project timeline
Sources: Finnveden et al., 2009; Zabalza-Bribián et al., 2011; Meex et al., 2020, Author, 2025

To close these gaps, many teams experiment with advanced data driven methods using machine learning models that can uncover complex relational patterns across geometry, material, and environmental datasets. Yet their usefulness is limited by “black box” opacity: stakeholders need not only accurate forecasts but also clear narratives linking input parameters to environmental outcomes. When algorithms cannot explain why a particular material palette or geometric strategy lowers embodied carbon, their recommendations are often distrusted or ignored (Doshi Velez & Kim, 2017; Molnar, 2022).

Moreover, the absence of holistic life-cycle visibility at the concept stage forces design teams to rely on rough heuristics or generic industry averages, rather than project-specific data. Architects may default to conventional, well-understood materials simply to avoid the uncertainty of unknown environmental variables, narrowing creative and sustainable options before alternatives can even be considered (Pomponi & Moncaster, 2017). This precautionary bias not only stifles innovation but also perpetuates status-quo material choices, undermining the very targets of circular economy frameworks. In practice, teams expend excessive time reverse-engineering EPDs or wrestling with incomplete manufacturer data, only to discover late in the process that their initial assumptions are invalid—triggering expensive and time-consuming redesigns (Gervasio & Dimova, 2018).

Core Problem

Architectural practice still lacks an integrated, explainable mechanism to quantify and communicate life cycle impacts during the conceptual stage, particularly for emerging materials. Solving this problem demands:

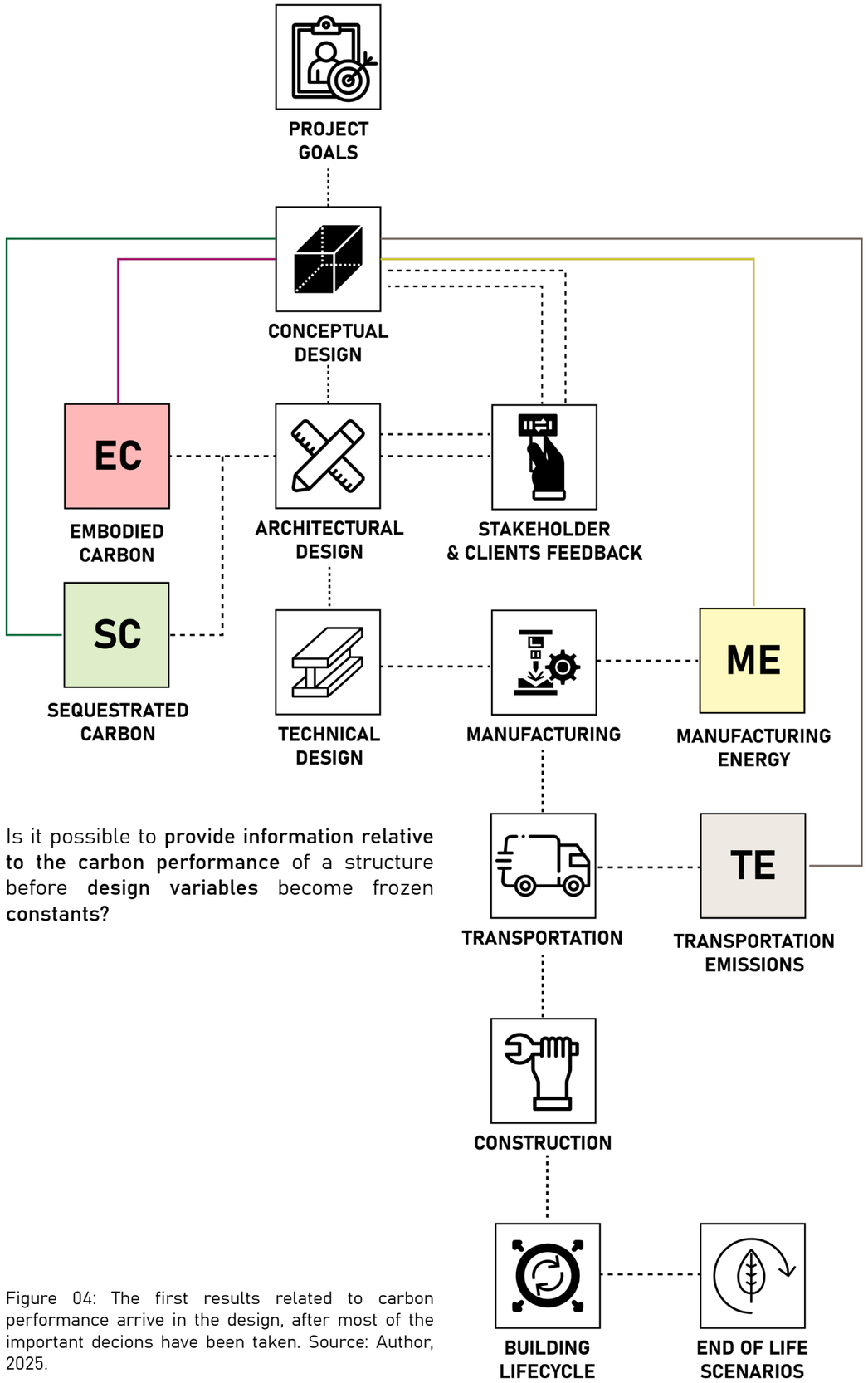
1.Embedding robust life cycle and material usage metrics in the very first design abstractions.

2.Enabling real time propagation of material and carbon data as the design evolves

3.Delivering transparent, actionable feedback that designers, engineers, and clients can interpret with confidence.

This research explores the potential of integrating live life-cycle metrics into the abstract inputs of early-stage design. It examines how each massing adjustment, opening placement or basic layout immediately triggers a chain of carbon and energy calculations cascading through manufacturing options, site logistics and end-of-life pathways (reuse, recycling, incineration or landfill). The study evaluates the nature and usefulness of the outputs produced and reflects on whether they provide sufficient insight to support informed decision-making. Preliminary findings suggest that this kind of integration can indeed lead to more informed design choices and reduce reliance on costly late-stage audits.

These elemental moves set off a domino effect from fabrication methods to delivery routes to disposal scenarios and surfacing their impacts in real time within the sketching environment transforms sustainability from an afterthought into the very language of ideation. Only when these conditions are met can circularity shift from retrospective objectives to proactive, design driving principles.



Is it possible to **provide information relative to the carbon performance** of a structure before **design variables** become frozen **constants?**

Figure 04: The first results related to carbon performance arrive in the design, after most of the important decions have been taken. Source: Author, 2025.

A. Introduction
A.3 Research Objectives

This thesis is driven by three primary objectives, each aimed at embedding life-cycle assessment into early-stage design through transparent, data-driven methods. These are supported by two sub-objectives that ensure practical implementation and validation.

Main Objectives

1.Create An Integrated Early-Stage LCA Framework
Develop a framework that integrates Environmental Product Declaration (EPD) data and structural configuration information with conceptual design inputs to generate reliable life-cycle and material-use metrics.

2.Develop And Calibrate Graph Neural Network (GNN) Models
Produce neural network models capable of estimating key life-cycle indicators for modular building components. These models will output rapid, component-level impact assessments that inform material and design choices at the conceptual stage.

3.Embed Explanation Mechanisms
To build user trust and support decision-making, this objective integrates interpretation techniques directly into the neural network workflow to quantify the contribution of each early design input.

Sub-Objectives

1.Create a Real-Time Feedback Interface
Deliver an interactive interface implemented as a plugin or extension for early-stage design that visualizes life-cycle and material usage metrics on demand.

2.Evaluate Scalability and Generalizability
Set up the framework so it can easily handle more materials beyond LVL and basic structural parts. Use simple, adaptable data formats and graph structures so adding new materials, mixed assemblies, or different project styles requires minimal changes.

A. Introduction
A.4 Research Questions

This section presents the principal and supporting research questions that guide this thesis. The first three questions address the core challenges of integrating life-cycle assessment into early-stage design, predicting impacts with machine learning, and delivering actionable feedback. Two additional sub-questions focus on ensuring transparency and extending the framework's applicability.

Main Research Questions

1.Early-Stage LCA Integration
How can a computational pipeline be designed to link Environmental Product Declaration (EPD) data and structural configurations so that reliable life-cycle and material-use metrics are available during conceptual design?

2.GNN-Based Impact Prediction
In what ways can graph neural networks be configured and trained to model building modules and their connections as graphs, and how accurately can they predict material throughput, and other key life-cycle indicators?

3.Transparent Insights
Which interpretability techniques (e.g., attention mechanisms, feature attribution) can be embedded in the GNN workflow to produce clear explanations of how specific design choices drive predicted material volume and life-cycle impacts?

Research Sub-Questions

1.Real-Time Design Feedback
How can the predictions generated by the GNN framework be visualized and integrated into early-stage design tools to support rapid iteration and informed decision-making by architects and stakeholders?

2.Framework Flexibility
What architectural strategies and data abstractions are needed to ensure the framework can be extended beyond LVL and structural elements to other materials, hybrid assemblies, and varied project complexities without extensive reconfiguration?

A. Introduction
A.5 Relevance

1.Immediate Value For The LVL-GNN Prototype

The core innovation is a working prototype that couples Laminated Veneer Lumber (LVL) product-specific Environmental Product Declarations with a graph-neural-network (GNN) surrogate.

Architects working on massing, layout and opening placement receive instantaneous estimates of embodied carbon, sequestered carbon, transport emissions and other key metrics. LVL—chosen for its high strength-to-weight ratio, dimensional stability and off-site prefabrication compatibility—is a flagship circular material yet is poorly represented in early-design tools (Abed et al., 2022).

By training the GNN on LVL assemblies, the system delivers not only sub-second predictions but also node-level attributions that explain precisely which user inputs drive the results.

2.Benefits For Industrial Partners And Practice

What formerly demanded hours or days of discussions or simulations now executes in fractions of a second. Designers gain immediate feedback before geometry freezes (cutting late-stage redesign costs), while suppliers acquire transparent, data-driven evidence of the competitive advantages of their bio-based components.

3.Explainable Machine-Leaning For The Building Environment

We formalize each building module as a graph node encoding all necessary information. The GNN will learn how these module characteristics, their connectivity patterns and aggregate counts shape material volume outcomes. Integrated attention mechanisms and feature-attribution techniques then pinpoint which specific building module attributes (e.g., opening size or count) most influence the prediction, directly addressing calls for interpretable AI in architecture and engineering (Doshi-Velez & Kim, 2017; Wu et al., 2021).

Sector-Wide & General Relevance

Construction already consumes 42 % of Europe's final energy, produces 35 % of its greenhouse-gas emissions, and extracts more than half of all raw materials, while up to 80 % of a building's lifetime impact is locked in during concept design. Embedding explainable LCA feedback at that stage therefore offers a uniquely high-leverage intervention that both fulfils the EU Circular-Economy Action Plan's requirement for design-stage carbon metrics and supports the rapid embodied-carbon cuts demanded by the Paris Agreement and IPCC AR6 mitigation pathways (Gervasio & Dimova, 2018; European Commission, 2020; Bodansky, 2016; IPCC, 2022).

A. Introduction
A.6 Research Scope

This thesis focuses on developing and validating for residential, low-rise buildings composed of prefabricated modules whose primary structural material is Laminated Veneer Lumber (LVL). A "building module" may constitute a single room or a cluster of rooms, and designs can aggregate one or multiple modules. To bound the design space, modules will follow a discrete size directory of 20 standard lengths and 10 standard widths with a fixed internal clear height of 3.155m.

Echo's performance metrics are drawn exclusively from European EPDs (in compliance with ISO 21930 and ISO 14025) and include:

-Global Warming Potential (GWPtotal & GWPbiogenic)
-Renewable / Non-renewable Energy Ratio (PERT / PENRT)
-Transportation Emissions
-Product Weight

While structural and thermal performance are guaranteed via externally developed, fully engineered LVL configurations (used to train the machine learning model), Echo itself remains strictly within the conceptual massing phase—infusing late-stage LCA data into early-stage decision making. The underlying dataset will be generated from an industry collaborator's 1:1, manufacturing-ready LVL solutions, ensuring that the neural network learns from real-world, engineered precedents.Although the tool is initially calibrated for European LVL products (e.g., LVL Q, LVL S), its data structures are designed for future expansion to other regions and materials. Echo covers both cradle to gate and cradle-to-grave scenarios, as reported in the source EPDs, enabling comprehensive life cycle insights during conceptual design (Bergman & Alanya Rosenbaum, 2017).

A.6.1 Boundaries & Conditions

Building Typology & Scale

1.Use Case: Residential, low-rise buildings (maximum of 2 floors).

2.Module Geometry: Standardized, rectilinear modules only; flat roof (no inclination or balconies). Dimensions drawn from a predefined directory: 20 length × 10 width each).

3.Module Count: Although the GNN training set includes designs with more than five modules to improve accuracy on larger assemblies, in practical use each design is limited to a maximum of five modules for manageability and computational efficiency.

Material Scope

1.Primary Material: Laminated Veneer Lumber (LVL) exclusively; no hybrid assemblies or multi-material integrations.

2.Manufacturer Criteria: Only products from EU-based manufacturers with fully transparent, publicly available Environmental Product Declarations (EPDs).

A. Introduction
A.6 Research Scope

3.Manufacturer Selection: EU-based producers with publicly available, fully transparent EPDs (e.g. Stora Enso, VMG Lignum, Metsa Wood). New entrants may be added provided their data quality and scope align with existing sources.

Design Phase & Methods

1.Conceptual Massing Workflow: In collaboration with Sustainer, each module is instantiated as a 1:1 3D box constrained by the size dictionary. All elements within the box carry unique identifiers and are organized in material-referenced layers.

2.Data Extraction: A processing script harvests key numerical parameters—module box dimensions, material volume after structural configuration, opening sizes and placements, and adjacency relationships between modules. Outputs are serialized as JSON files.

3.GNN Input Construction:JSON descriptors encode box size, layout, opening placement, and interface patterns. The trained network uses these descriptors to predict material volumes and environmental metrics without accessing confidential 3D geometry.

4.Decoupling from Proprietary Models: The final tool relies solely on derived JSON data; there is no runtime connection to Sustainer's internal 3D files.

Performance Metrics

All metrics adhere to **ISO 21930** and **ISO 14025** standards and derive directly from manufacturer EPDs.

1.GWPtotal (Global Warming Potential, total): Sum of all greenhouse gas emissions (including biogenic and fossil) expressed in CO_2-equivalents over the product's life cycle stage.

2.GWPbiogenic (Global Warming Potential, biogenic): CO_2-equivalents associated with biogenic carbon uptake and release (e.g., from wood growth and decay).

3.PERT (Primary Energy Renewable Total): Total primary energy from renewable sources (MJ).

4.PENRT (Primary Energy Non-Renewable Total): Total primary energy demand from non-renewable sources (MJ).

5.Transportation Emissions: CO_2-equivalent emissions calculated from actual transport distance between LVL manufacturing site and project location. Routing uses OSM-based truck paths and includes ferry segments when required. Emission factors harmonized across manufacturers.

6.Mass: Total dry mass of LVL modules per EPD data (kg).

Life Cycle Stages

1.Cradle-to-Gate (A1-A3): Raw material extraction through LVL manufacturing up to factory exit.

2.Cradle-to-Grave (A1-C4): Includes use phase and end-of-life processes where available.

3.Cradle-to-Grave & Beyond (A1-D): Extended scenario capturing demolition and post-demolition stages when EPDs include those phases.

End-of-Life (EoL) Scenarios

1.Primary Scenario: Incineration (energy recovery) as the main EoL pathway.

2.Secondary Scenarios: Recycling (LVL recovery), Reuse (LVL components repurposing), and Landfill.

3.EPD Variability: Where manufacturers differ in EoL assumptions, scenarios are modeled separately and results presented distinctly.

Geographic Context

1.Region: European Union (EU) data and regulatory context.

2.Future Extensibility: Framework designed to accommodate non-EU regions in later research by substituting equivalent EPD datasets.

Validation

1.Precedent Dataset: A curated library of fully engineered, full-scale (1:1) LVL designs.

2.Neural Network Training: Models trained on precedent dataset to predict performance in new conceptual massing scenarios.

Exclusions & Out-of-Scope Items

1.Interior Quality: No interior design or indoor environmental quality metrics included.

2.Economic & Social Indicators: Cost analysis, embodied water, toxicity, and other non-EPD metrics are excluded.

A. Introduction
A.7 Industry Collaboration

To ground the Echo prototype in real-world workflows, the research is partnered with Sustainer—a Dutch construction-tech innovator founded in Utrecht in 2014 and accelerated through Startupbootcamp. Their flagship Home platform delivers high-tech, prefabricated LVL modules optimized for structural performance, acoustics, fire safety, and design flexibility. Deployed in nearly 100 permitted projects across 30 municipalities, Sustainer brings both domain expertise and a proven production pipeline.

What Sustainer Provides for the research

Rhino Plugin Access: Their Rhinoceros plugin allows for the transformation of simple 3D boxes to fully realised building modules containing all the necessary building components in 1:1 level of detail ready for manufacturing.

Specifically:

1.Module Definition: A built-in menu lets users choose from Sustainer's library of standard module sizes, placing box-shaped geometries directly into the conceptual model.

2.Opening Placement: Doors and windows are represented as closed-curve placeholders, which users can position freely on any module face.

Two-Stage Geometry Generation

Basic Level: Enables rapid massing via simple geometric proxies and curve controls.
Construction Level: On demand, the plugin transmits module dimensions, opening coordinates, and the selected building system to Sustainer's cloud server. The server returns full-scale, manufacturing-ready LVL models—complete with connection details and annotations—streaming them back into Rhino.

Figure 05: Sustainer Homes has launched the first self-sufficient house, based on modular timber-frame construction. The young team of founders offers an all-in-one concept from design through delivery.
Source:https://www-architectuur-nl.tudelft.idm.oclc.org/nieuws/sustainer-home-modular-gelanceerd

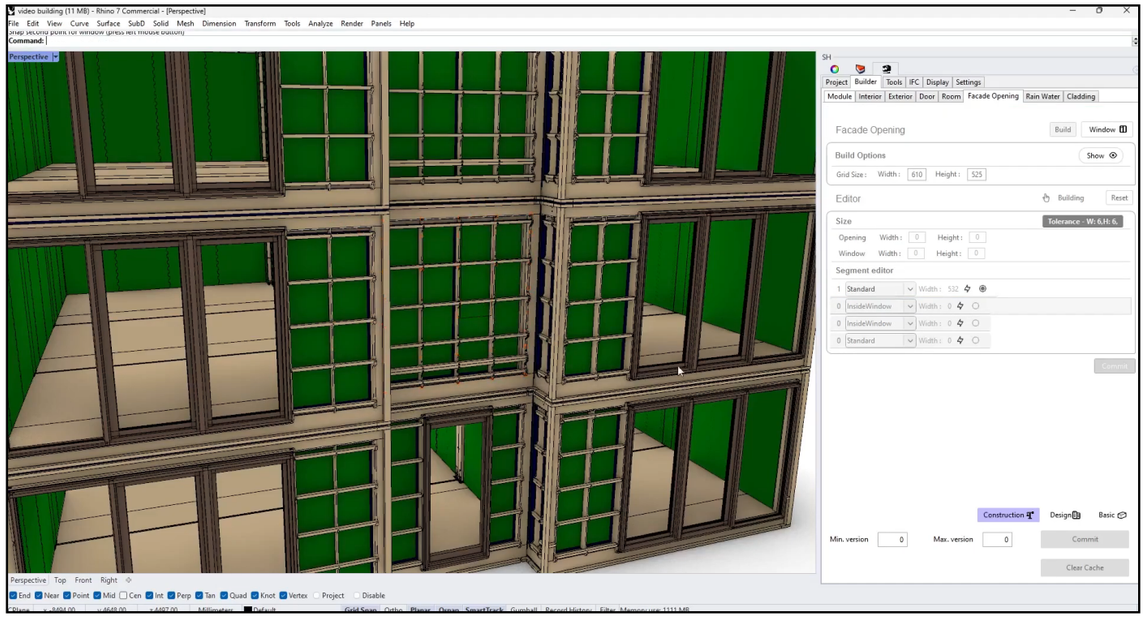


Figure 06: Their toolkit is in the form of a Rhinoceros plugin, where the design input from the users computes all the necessary changes in the structural configuration.
Source: Sustainer's YouTube Channel / https://www.youtube.com/watch?v=N0bEANU7B8s&ab_channel=Sustainer

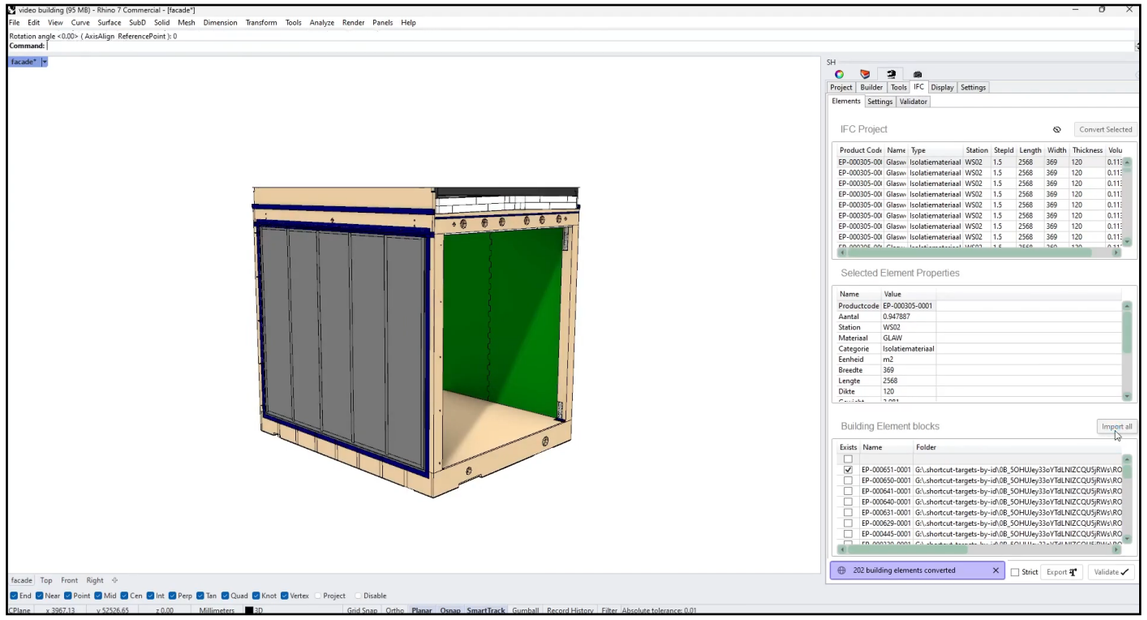


Figure 07: With Sustainer's assistance, the user can generate a building module just by referencing a 3D box. Then the user can isolate any object/element of the module to inspect.
Source: Sustainer's YouTube Channel / https://www.youtube.com/watch?v=N0bEANU7B8s&ab_channel=Sustainer

A. Introduction
A.8 Research Methodology

A.8.1 Work Packages

The Echo project unfolds across five interdependent work packages (WPs), each building on the last to deliver a rigorously validated, academically robust, and industry-aligned design tool.

WP1: Literature Review & Requirements Gathering

WP1 establishes the project's theoretical and practical footing through an extensive program of literature review and requirements gathering. It synthesizes insights from complementary fields:

1. The granular level of detail in architecture
2. Circularity in the built environment
3. Laminated veneer lumber and modular design
4. Digital tools for circularity
5. Graph neural networks

This research assists us not only in identifying important life-cycle indicators but what are the existing variables during the conceptual stage, which can influence them. Guided by these findings, WP1 formulates the core requirements for an early-stage life-cycle assessment and material-usage tool: it extracts actionable circular-economy metrics suited to schematic decisions, distils interface principles that inform rather than dictate choices, evaluates state-of-the-art relational neural architectures, and defines input–output specifications that avoid specialized expertise and therefore broaden accessibility. The work culminates in three deliverables:

1. Literature Review that addresses the research questions and objectives.
2. Data Requirements Specification Sheet
3. Bibliography Of Scientific Sources

WP2: Data Collection & Processing

Work Package 2 (WP2) translates dispersed technical sources into structured, machine-readable datasets for the project's graph-neural-network (GNN) design assistant. Third-party-verified Environmental Product Declarations (EPDs) for European laminated-veneer lumber (LVL) are harmonized to capture global warming potential, cumulative energy demand, transport emissions, and mass. All parameters are extracted through the International EPD System and entered into an Excel template that later serves as the post-processing layer for GNN outputs, linking each metric to one cubic meter of LVL—and, by extension, to the corresponding design instance.

In parallel, WP2 develops a geometry-to-feature pipeline that converts every LVL module into a graph with node, edge, and global attributes, preserving the relational information required for GNN training and inference. The pipeline translates each design into a graph in which every module appears as a node annotated with readily available early-stage attributes that describe its own characteristics and interfaces with neighboring modules. This process takes the 3D geometry information provided by Sustainer, and each design participating in the training and validation dataset is converted into a JSON data structure format that holds all the information for the graphs in numerical format.

Data undergoes checks for completeness, consistency, and correct units, and a representative sample of design scenarios is selected, using Latin Hypercube Sampling to keep the number of combinations manageable.

Two straightforward results come out of WP2:

1. Graph dataset for the GNN. Every design is turned into a JSON-based graph (nodes, edges, feature vectors) that the neural network can train on and query.
2. Clean EPD lookup table. An Excel sheet of LVL environmental data (carbon, energy, transport, mass) that plugs straight into the model's outputs to translate graph results into real-world impacts.

WP3: Graph Neural Network, Training & Evaluation

Work Package 3 (WP3) transforms WP2's curated JSON design files into a predictive engine that estimates, for every module in a design, the number of LVL objects it should contain and the volume of material it will use.

The workflow begins by validating each file and turning its content into a mathematical graph whose nodes represent the individual modules and whose edges record their physical connections. Those graphs are packaged into two tensors: one stores the input attributes per node, and the other stores the two target values the model must learn.

The main goal of WP3 is to ensure that the model can predict the LVL Volume of each module in a design, ensuring an accuracy of 85% while training on the dataset. After training, the model is further validated using a set of designs with highly increased complexity compared to the dataset the model used to train on, but with the same graph structure. It is natural at that point to see a further loss in accuracy, but the model needs to be structured so that we do not see a level of accuracy of less than 75% to preserve proof of concept.

During development and training, it is crucial to implement methods to evaluate the model's results to understand how the input features, hence the user input, contribute to the targeted values the neural network is trying to predict. To achieve that, we deploy a set of visualizations to try to interpret and evaluate its performance, like monitoring the accuracy during every epoch to avoid overfitting or underfitting, and most importantly, the integrated gradient attribution maps, where we measure how much every input feature contributes to the predictions.

Building on the dataset from WP2, WP3 iteratively designs and evaluates graph neural networks. Through cyclical training, testing, and refinement, the work package balances predictive performance with computational efficiency and incorporates interpretability mechanisms to foster stakeholder trust.

Deliverables

1. Model Prototype: Trained GNN model with documentation.
2. Trained model parameters. Store the trained model in a pth. file for further use in other work packages like WP4.

A. Introduction
A.8 Research Methodology

WP4: Tool Integration & Development

WP4 embeds the refined GNN model into a Grasshopper plugin, delivering real-time feedback on life-cycle metrics during conceptual layout adjustments. Usability testing and close collaboration with model development ensure that the interface remains intuitive and that prediction quality meets design needs. All the research and development of Echo comes down to a series of Grasshopper components where designers can freely use in real time to assess the LCA performance of their conceptualization and discuss with stakeholders and clients for possible alterations.

Deliverables
1.Grasshopper Components
a) Import Project. This component connects to the EPD lookup table for WP2 while also prompting the user to choose between different LCA scenarios, like cradle to gate/grave or end of life. The user can also inform the system of the project's location, optionally to get feedback on later transportation.
b)Material Calculator. This component is the primary source of information. It loads the trained GNN model from WP3 and starts running in real time to provide calculations focused on the targeted metrics. The plugin requests information from the user output from the "Import Project" component and nothing else, since it is linked directly with the layers found in Rhino and updates the calculation when the user alters their design.
c)Holistic LCA. Tool to analyze the performance of each module in a design. The component creates a spider chart presenting the targeted metrics of each module, the users can see how each module performs while they design.
d)Design Summary. Tool to print out all the information for a design per building module.
e)Carbon Performance. This component creates a bar chart displaying the carbon performance either per building module or the whole design. By tweaking the design parameters and going through the different lca and eol scenarios, the use can assess the design.
f)Feature Contribution. The tool outputs how much each of the user's input contributes to the resulted material volume.

WP5: Research Outcomes & Contributions

WP5 serves as the culmination of the project's research. It synthesizes the cumulative findings from WP1 through WP4—spanning literature review, data collection, GNN model development, and real-time interface integration—and translates prototype performance and case-study validations into a comprehensive evaluation of research outcomes and the framework's scalability potential.

As the final report, WP5 collates all quantitative metrics and qualitative insights generated throughout the project—model predictive accuracy, explanation fidelity, user-interface usability, and industrial partner feedback—to present a clear, cohesive narrative of how the integrated LCA framework meets the research objectives. WP5 examines scalability and generalizability. It assesses the adaptability of the Echo pipeline to alternative material systems, building typologies, and geographic contexts. This section will identify both technical enablers, such as modular data structures and pluggable EPD integrations and key barriers like data availability and domain-specific calibration requirements.

A. Introduction
A.8 Research Methodology

A.8.2 Project Timeline

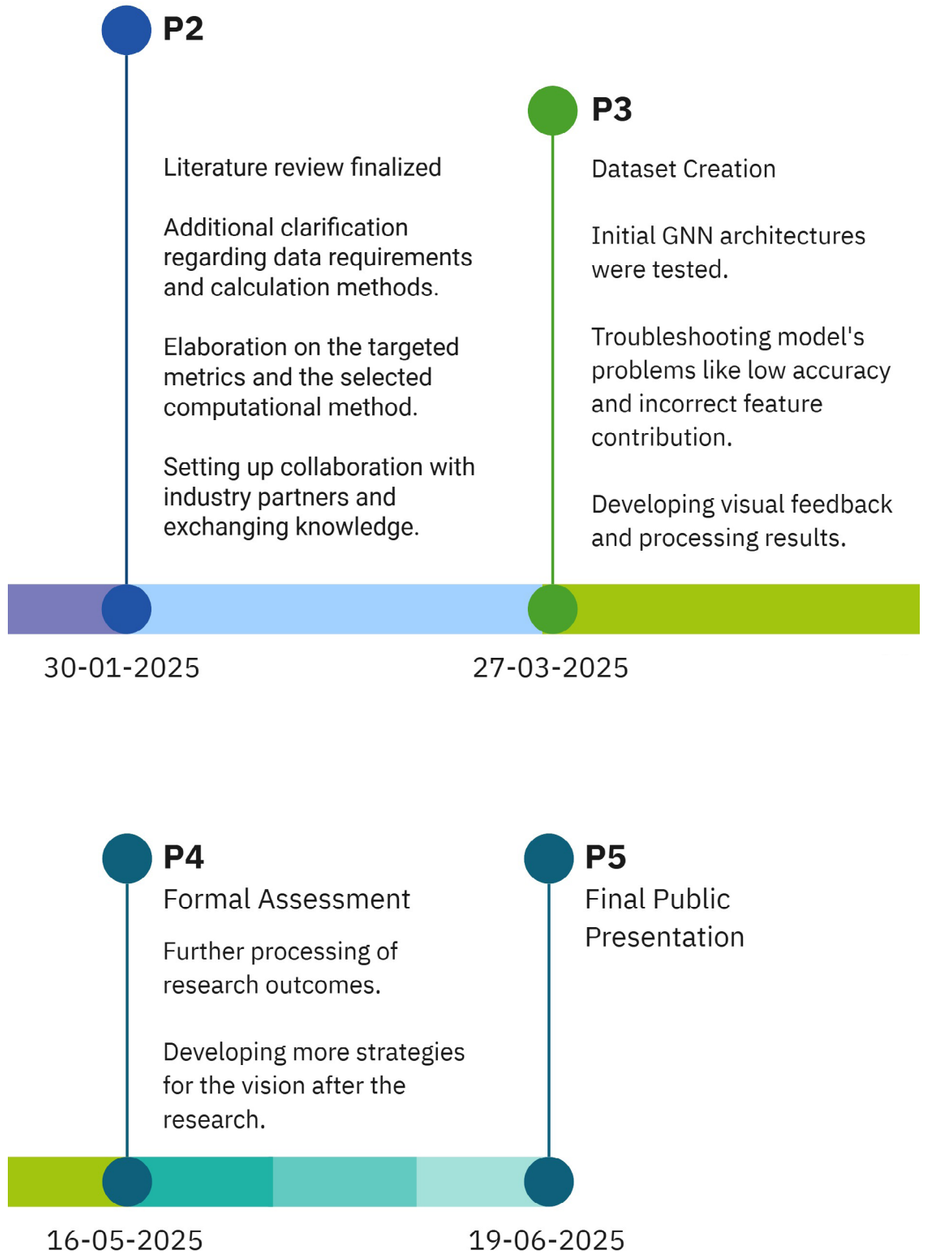


Figure 08: Research Timeline / Source: Author, 2025.

A. Introduction
A.8 Research Methodology

A.8.4 Workflow Diagram

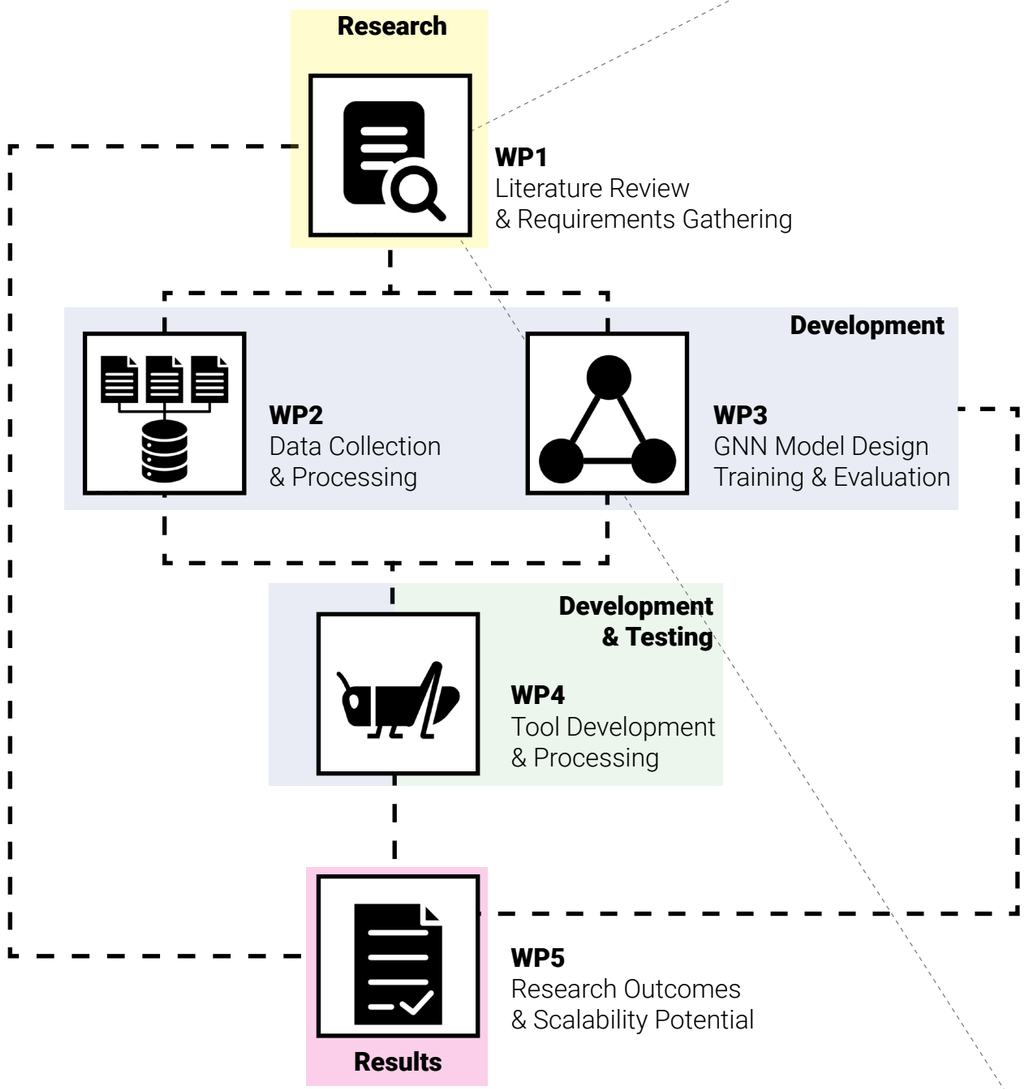


Figure 09: Work Packages Overview / Source: Author, 2025.

In this section, we begin with a concise, high-level overview of the entire project. We then dive into each work package,explaining visually the output and key deliverables and methodological approach. For every package, we highlight how it feeds into and depends on the others, illustrating the network of interdependencies that ensures coherence and drives progress.

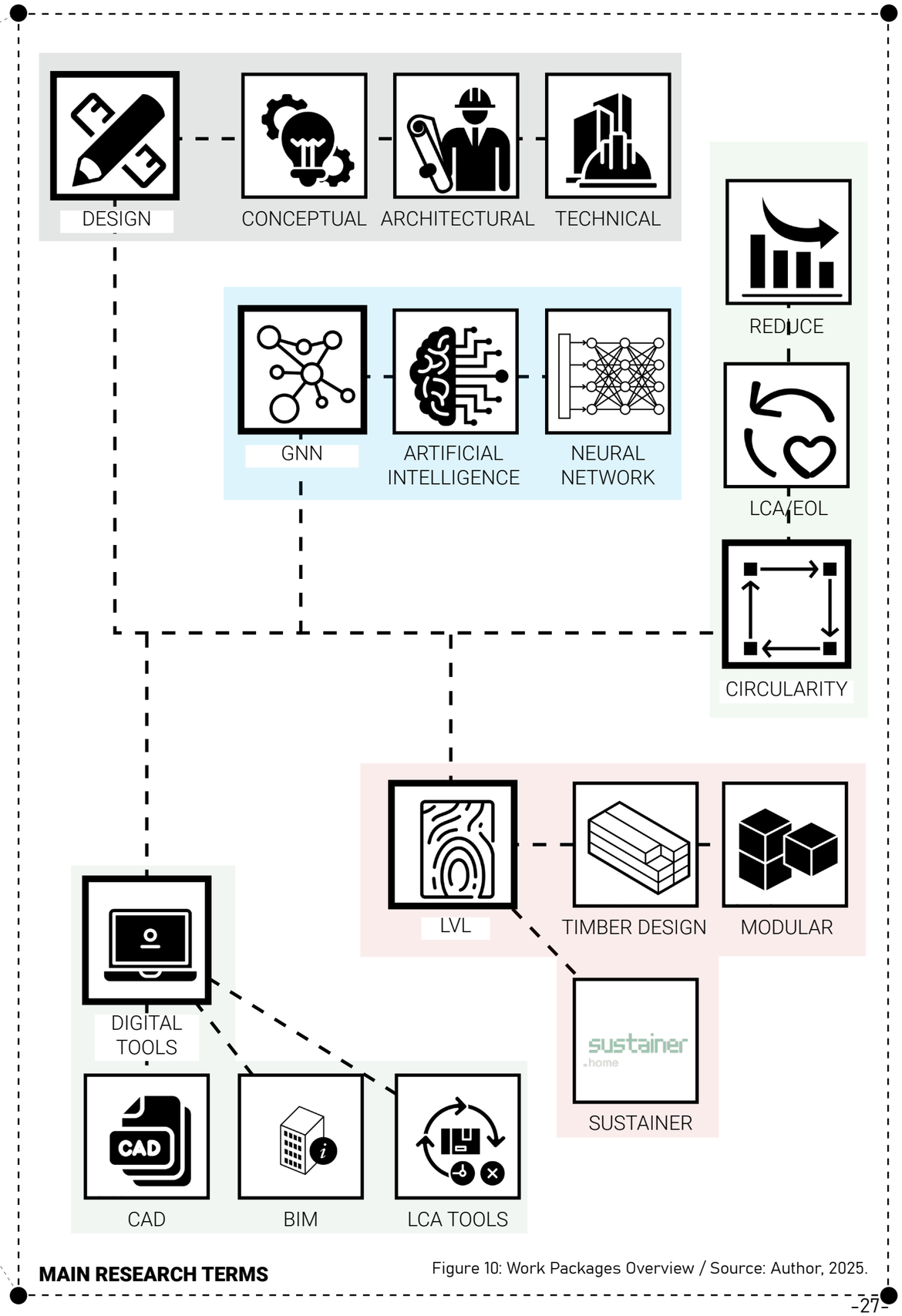


Figure 10: Work Packages Overview / Source: Author, 2025.

A. Introduction
A.8 Research Methodology

A.8.4 Workflow Diagram

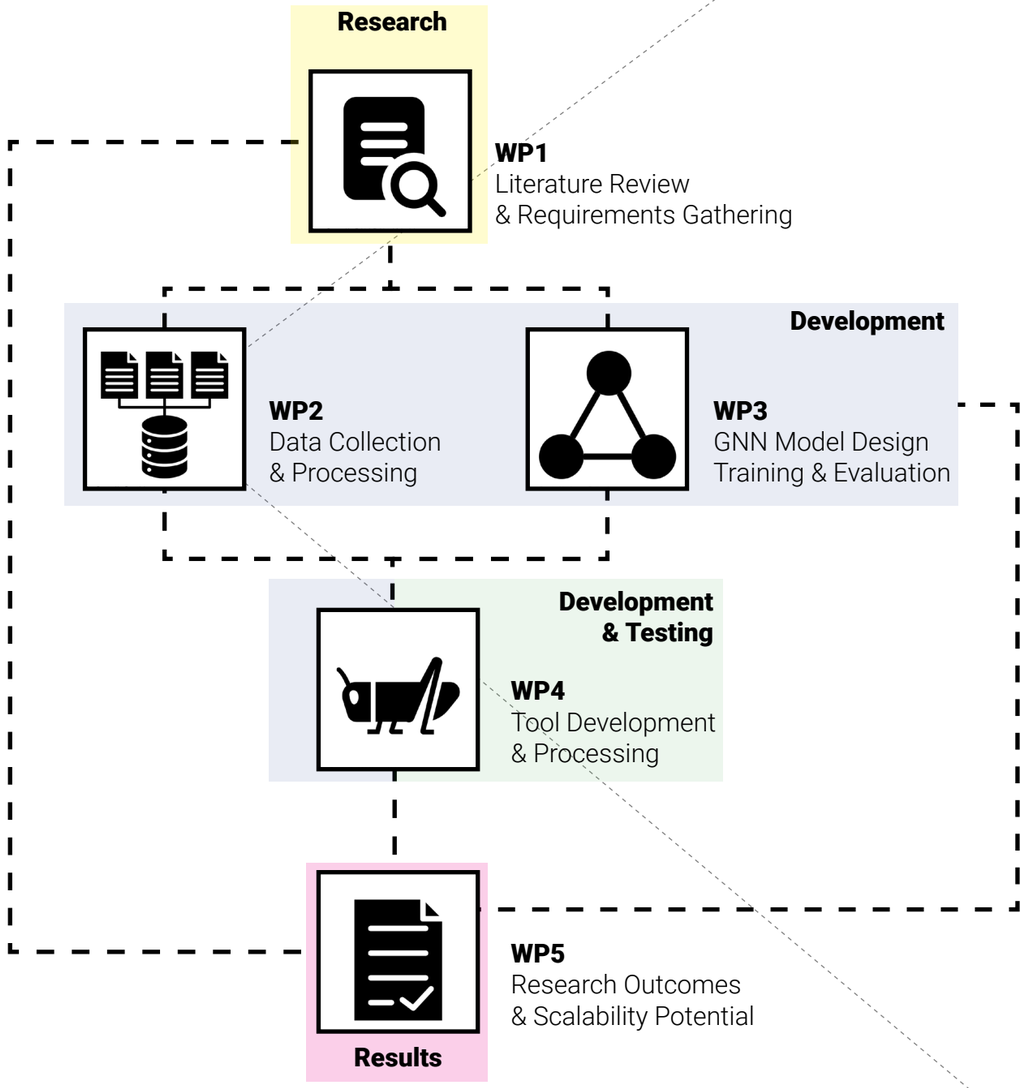


Figure 09: Work Packages Overview / Source: Author, 2025.

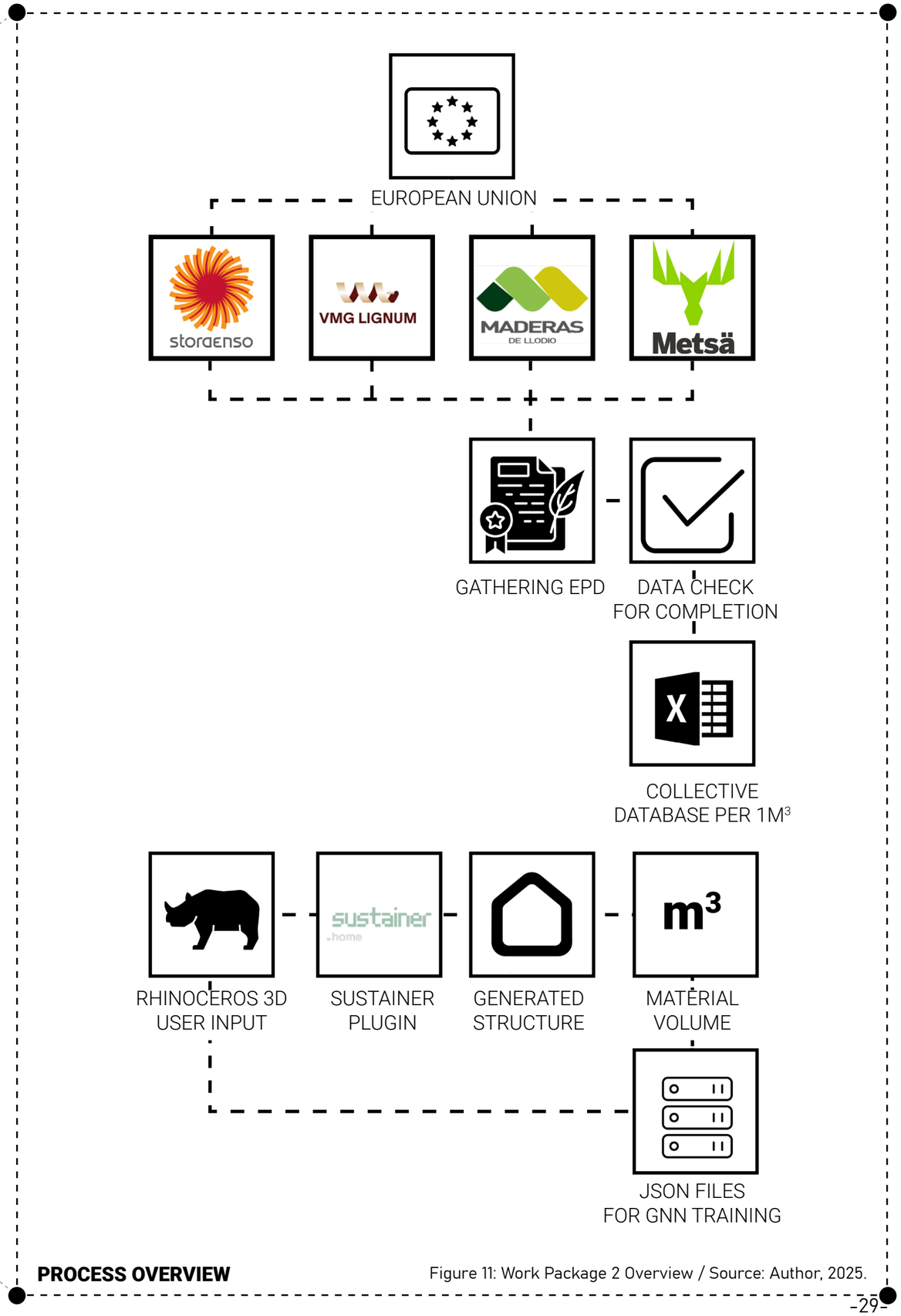


Figure 11: Work Package 2 Overview / Source: Author, 2025.

A. Introduction
A.8 Research Methodology

A.8.4 Workflow Diagram

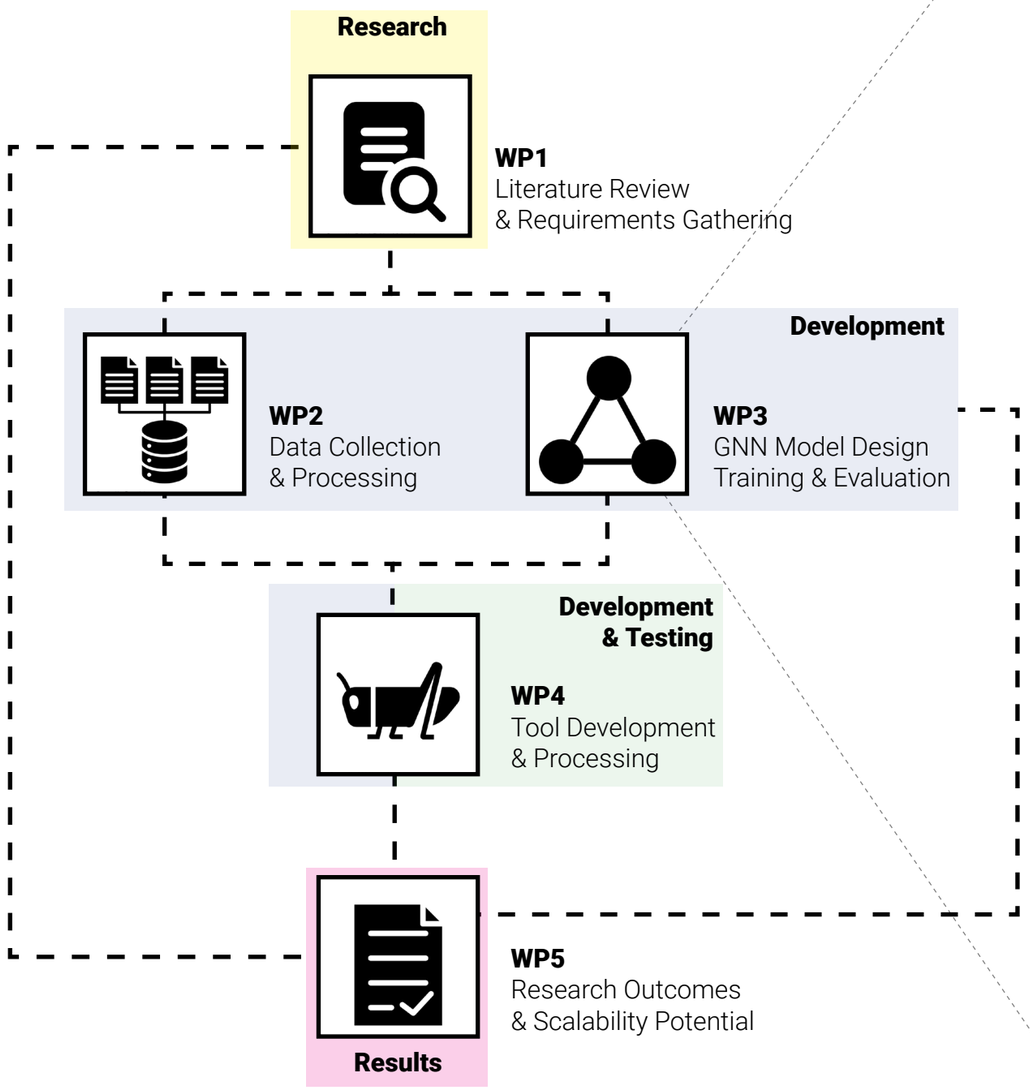


Figure 09: Work Packages Overview / Source: Author, 2025.

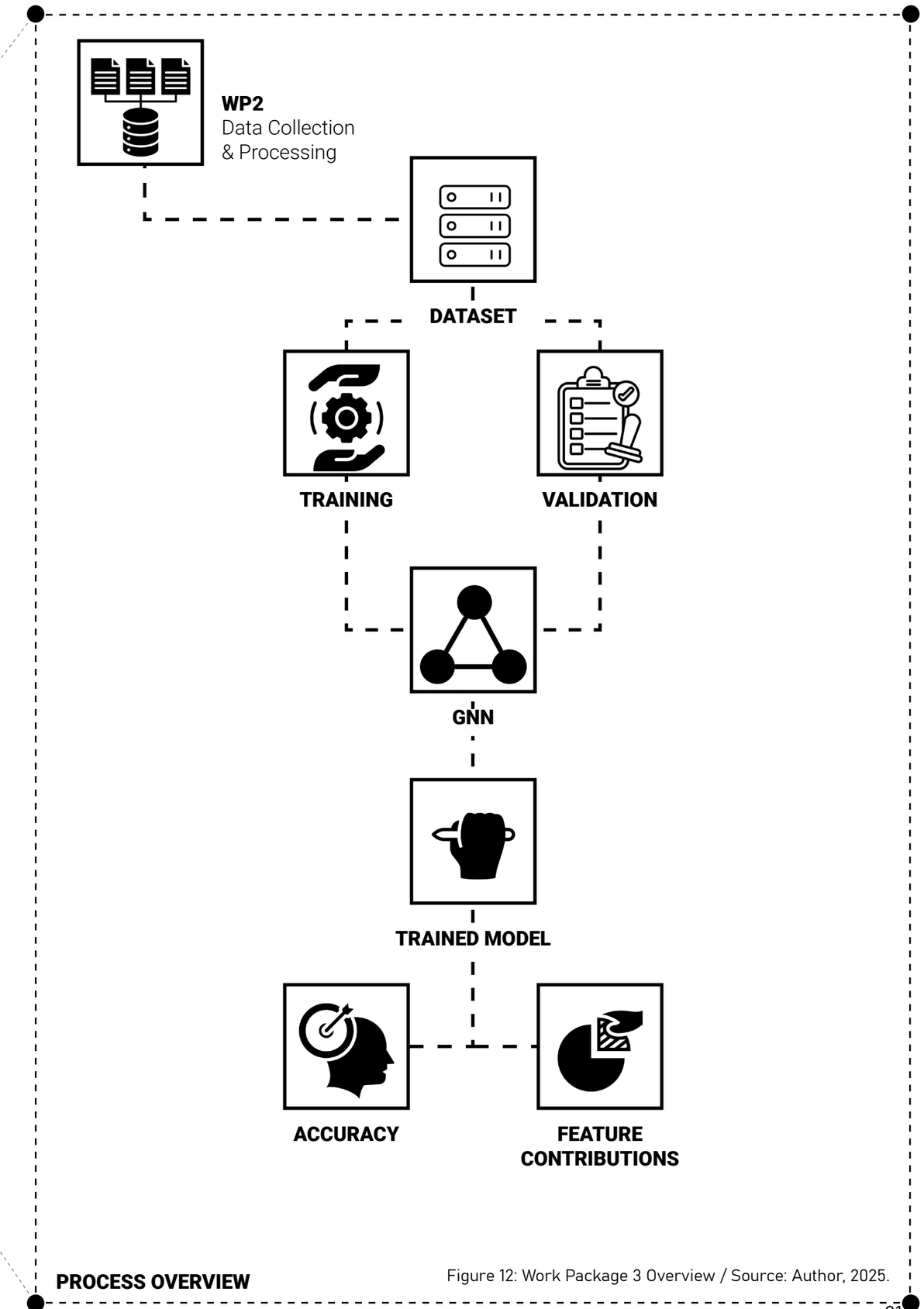


Figure 12: Work Package 3 Overview / Source: Author, 2025.

A. Introduction
A.8 Research Methodology

A.8.4 Workflow Diagram

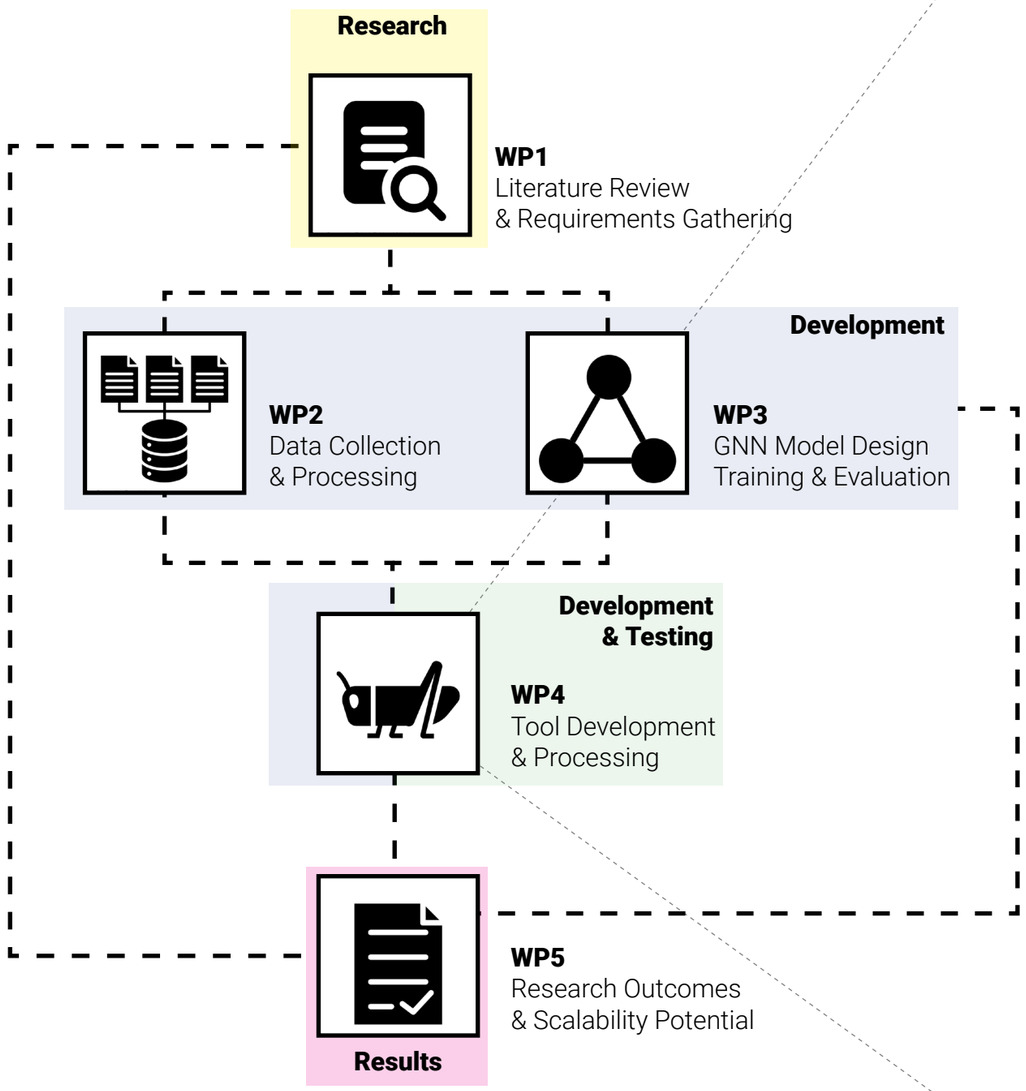


Figure 09: Work Packages Overview / Source: Author, 2025.

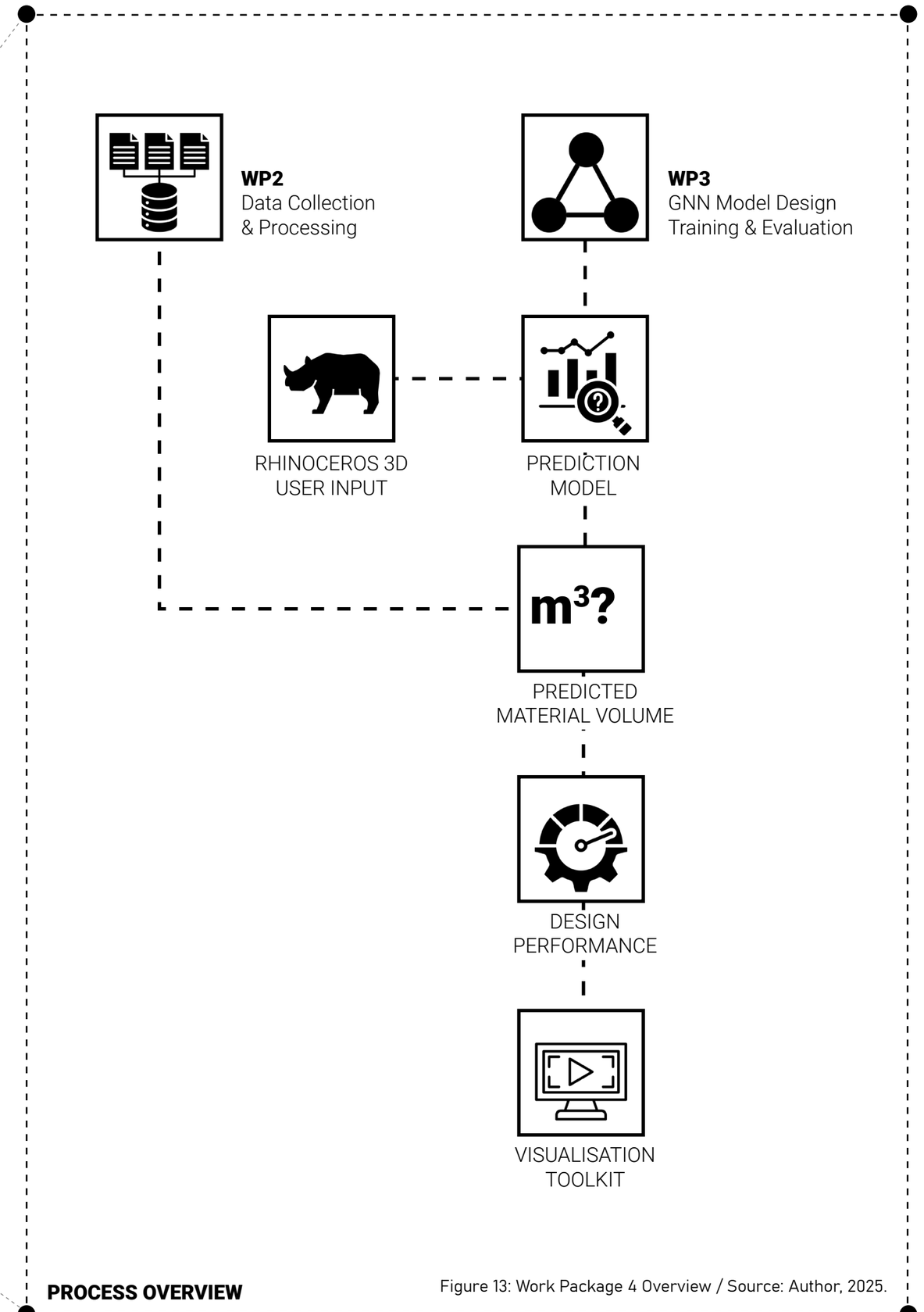
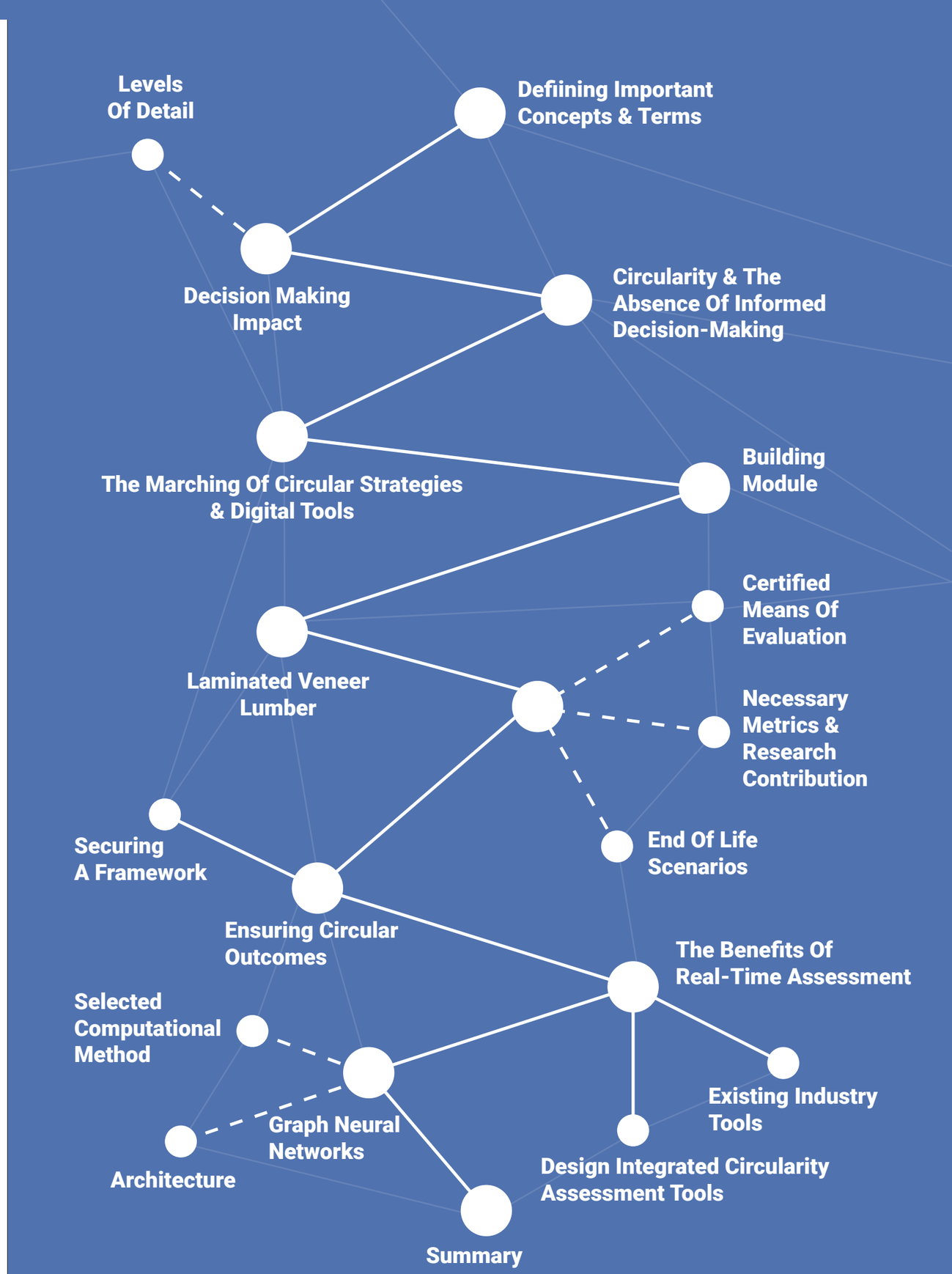


Figure 13: Work Package 4 Overview / Source: Author, 2025.

Work Package 01 | Literature Review & Requirements Gathering



B. Literature Review

Brief Summary

Chapter B establishes the foundational concepts and terminology necessary for integrating circular economy principles into architectural design. It begins by defining the circular economy and its “reduce” strategy, and explains life-cycle assessment as the method for quantifying environmental impacts from raw materials through end-of-life (Kirchherr et al., 2017; Finnveden et al., 2009). The review emphasizes the importance of planning for reuse, recycling, energy recovery or landfill to preserve material value (Ghisellini et al., 2016) and introduces environmental product declarations as verified sources of product-level impact data (Del Borghi, 2013). It then describes digital design tools—from BIM and parametric workflows to virtual reality—and contrasts rule-based systems with computational intelligence techniques like neural networks, highlighting the interpretability challenges of black-box deep learning (Adadi & Berrada, 2018; Montavon et al., 2018). By examining how conceptual, architectural and technical design phases differ in data granularity, the chapter reveals where real-time, interpretable feedback is most needed (Dym et al., 2005; Schuster & Geier, 2022). It shows how modular prefabrication with laminated veneer lumber can embody circularity in practice (Abed et al., 2022; Bergman & Alanya-Rosenbaum, 2017) and surveys existing LCA frameworks and end-of-life scenarios. Finally, it identifies graph neural networks as a unified, explainable approach capable of carrying spatial, environmental and manufacturing data in a single model, delivering rapid, transparent life-cycle feedback during early design.

B. Literature Review

B.1 Defining Important Concepts & Terms

Before advancing to more specialized research topics related to computation, design, and circularity in architecture and the built environment, it is essential to provide some high-level information on foundational concepts and terms.

1.Circularity In The Built Environment & The Strategy Of Reduce

A circular economy is "an economic system that replaces the end of life concept with reducing, A circular economy is "an economic system that replaces the end-of-life concept with reducing, alternatively reusing, recycling and recovering materials in production/distribution and consumption processes" (Kirchherr, Reike & Hekkert, 2017). It emphasizes designing out waste and pollution, keeping products

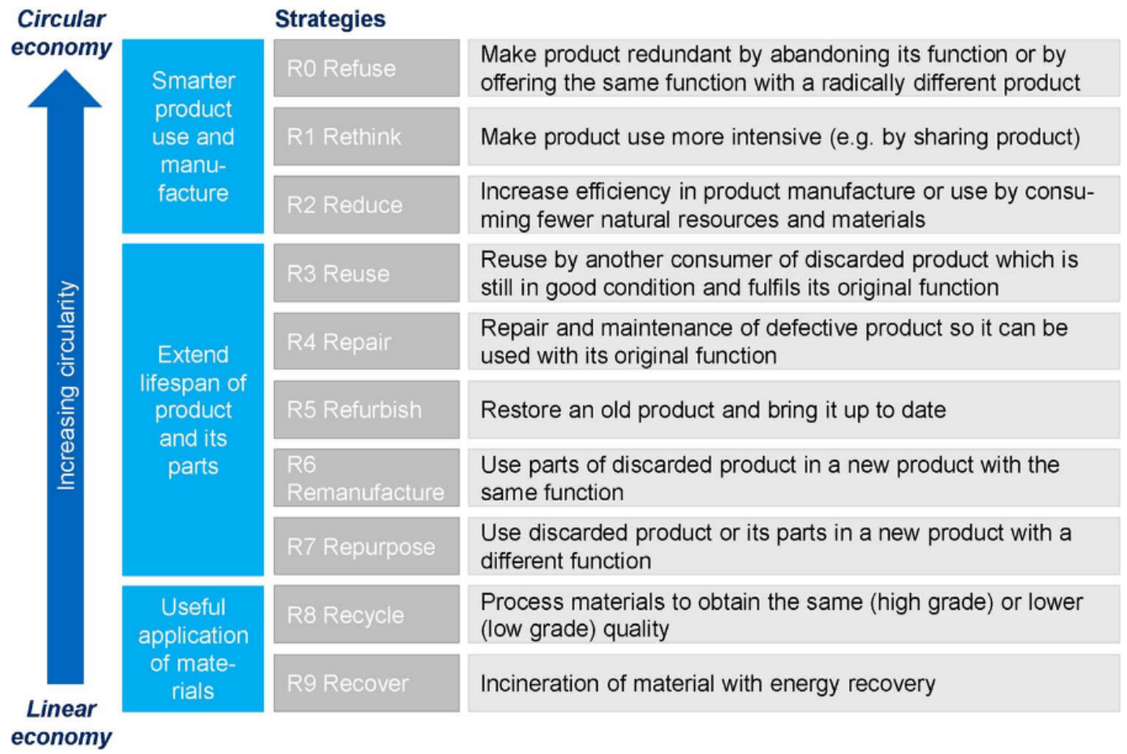


Figure 14: The 9R Framework. From Conceptualizing the circular economy: An analysis of 114 definitions by J. Kirchherr, D. Reike, and M. Hekkert, 2017, Resources, Conservation & Recycling, 127, p. 5. https://doi.org/10.1016/j.resconrec.2017.09.005

and materials in use, and regenerating natural systems.

The "Reduce" strategy in circular-economy-driven architecture and construction prioritizes minimizing material inputs and waste at the outset of a project's life cycle, thereby delivering substantial environmental and economic benefits. By integrating design optimizations—such as precise material quantification and modular prefabrication—the sector can curb resource depletion, lower greenhouse-gas emissions, and conserve landfill space, while also enhancing building durability and market value (Rahla et al., 2021).

Why an up-front material overview is critical

A.Live quantity take-offs and carbon dashboards
Digital design platforms can generate a real-time bill of materials the moment a wall thickness changes or a column is shifted. Designers can instantly see the mass of concrete, kilograms of steel, and corresponding embodied carbon, each choice locks into the project.

B.Early substitution of high-impact components
With quantities quantified, designers can swap high-carbon products (e.g., virgin aluminium façades) for lower-impact alternatives (e.g., recycled-content aluminium, timber, or bio-based composites) while still on the drawing board, avoiding costly redesigns later.

C.Right-sizing prefabricated and modular elements
Knowing exact dimensions and counts enables factories to fabricate components to the millimetre, reducing off-cuts, over-ordering, and transport volumes. Modular approaches also simplify future disassembly and reuse, extending the life of every kilogram brought onto site.

D.Quantified, verifiable carbon savings
Because the Reduce strategy starts with counting materials, the carbon impact of decisions can be reported with confidence to clients, regulators, and investors. This evidence base strengthens the business case for circular construction, turning sustainability commitments into measurable performance indicators.

3.Life Cycle Assessment (LCA)

Life Cycle Assessment is a standardized method to quantify environmental impacts and resource use throughout a product's life cycle, from raw material extraction through production, use, and end of life management (Finnveden et al., 2009). LCA provides the data foundation for informing sustainable design and procurement decisions.

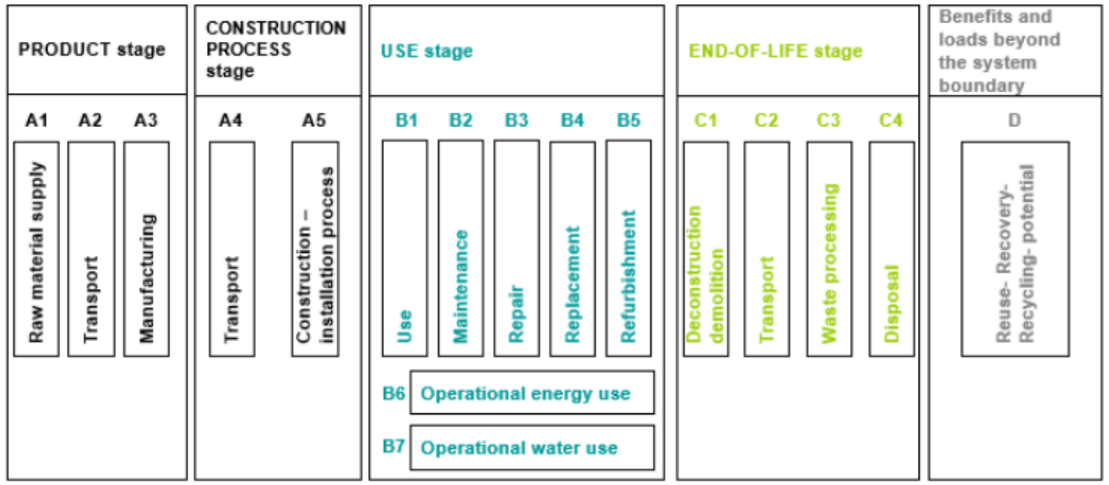


Figure 15: The Life Cycle Phases European Commission, Joint Research Centre, 2018 (EUR 29145 EN). Publications Office of the European Union. Retrieved from https://op.europa.eu/en/publication-detail/-/publication/45a30a5c-3169-11e8-b5fe-01aa75ed71a1/language-en

B. Literature Review
B.1 Defining Important Concepts & Terms

4.End-of-Life (EoL)

End-of-Life refers to the final phase in a product's life cycle when a building component or material reaches the conclusion of its useful service life and is subject to disposition strategies such as reuse, recycling, energy recovery, or landfilling (Ghisellini, Cialani, & Ulgiati, 2016, DOI:10.1016/j.jclepro.2016.12.048). **Within circular-economy frameworks, EoL planning seeks to preserve material value and minimize environmental impacts by selecting pathways that enable material recirculation**

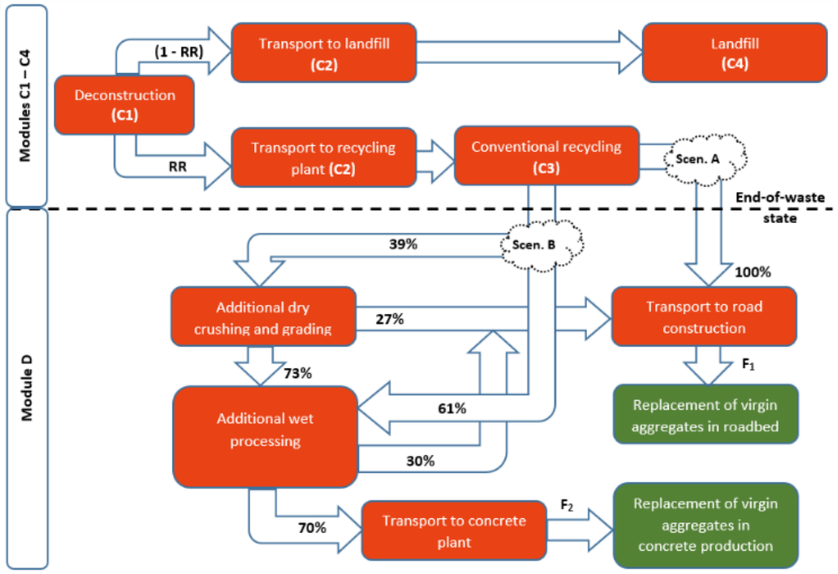


Figure 16: EOL Pathways for concrete from LCA and Consumption/Needs-Based GHG Accounting for Climate Action: A Pathway to Carbon Neutrality (p. 5), by K.-G. Kim, 2025, Springer Nature Switzerland AG. https://doi.org/10.1007/978-3-031-75468-5

4.Environmental Product Declarations (EPDs)

Environmental Product Declarations (EPDs) are Type III environmental labels that communicate quantified environmental information based on a product's LCA, following ISO 14025 procedures and verified by independent third parties (Del Borghi, 2013; ISO, 2006). EPDs enable transparent, comparable assessments of products' cradle to grave impacts.

5.Digital Design Tools

Digital design tools have fundamentally redefined the architect's workflow by embedding "digital thinking" into every phase of a project's life cycle. Beyond mere drafting software, these tools encompass computer-aided design for precise 2D and 3D geometry, building information modeling for data-rich, parametric objects that facilitate coordination and documentation, and geographic information systems for contextual site analysis. Simulation and analysis applications, ranging from energy performance and daylighting assessments to preliminary structural feedback, enable architects to make informed, sustainability-driven decisions without requiring deep specialist expertise (Ma, Azari, & Elnimeiri, 2024).

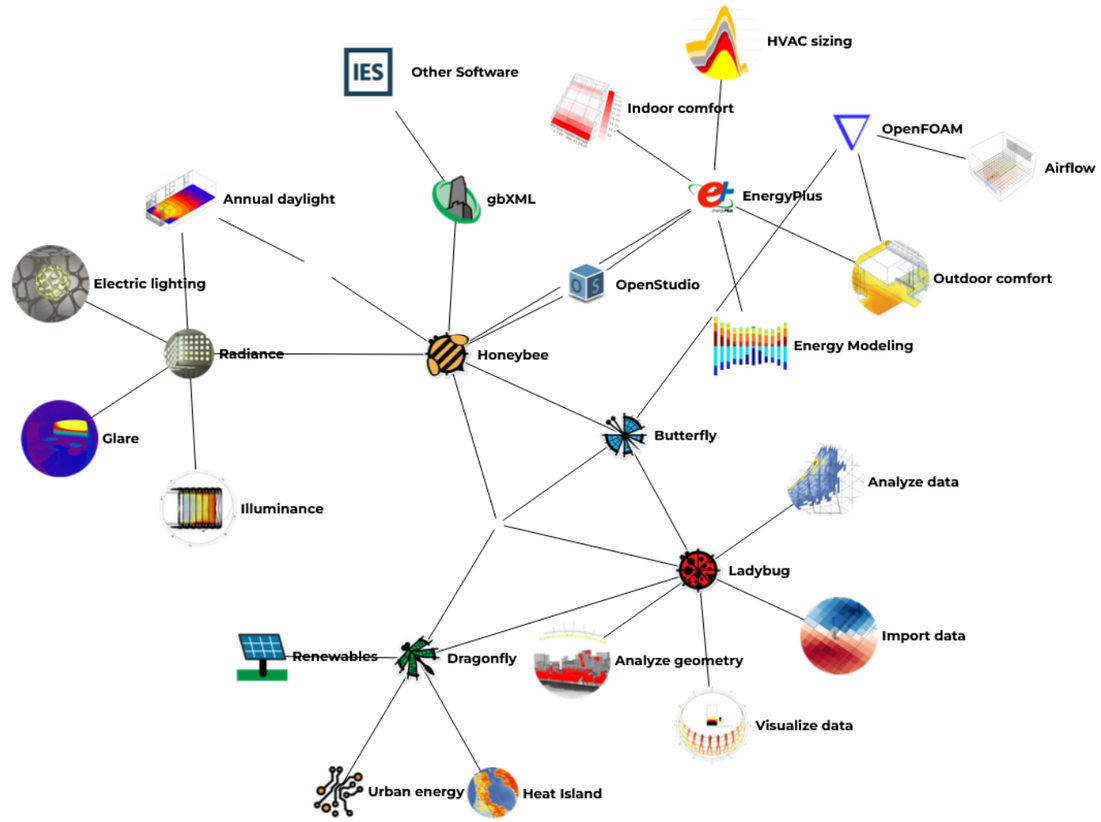


Figure 17: Environmental tool network in Grasshopper from https://www.ladybug.tools/honeybee.html

6.Computational Intelligence And Machine Learning

Computational intelligence (CI) excels at the complex, uncertain, and ill defined challenges of architectural design and building operations. Designers must juggle many constraints, predict performance under changing conditions, and accommodate human behavior that defies precise equations. **CI techniques, fuzzy logic, neural networks, and evolutionary algorithms, embrace this ambiguity: fuzzy logic captures gradations like "low," "medium," and "high" daylight with simple if then rules; neural networks reveal hidden links between layout choices and outcomes such as energy use or circulation patterns (Novedge, 2024);** and evolutionary algorithms sift through thousands of variants to find optimal trade offs, such as bright rooms with minimal energy demand. **Machine learning drives these CI methods, marking the shift from rule based expert systems (e.g., MICON, VEXED) to data driven forecasting, classification, clustering, pattern recognition, and generative design (Reddi & Yazdanbakhsh, 2025; Özerol & Arslan Selçuk, 2023).**

B. Literature Review
B.1 Defining Important Concepts & Terms

7.Neural Networks And The Black Box Phenomenon

Artificial Neural Networks (ANNs), often simply referred to as neural networks, are computational models inspired by the human brain's interconnected neurons, enabling systems to learn patterns and relationships directly from data (LeCun et al., 2015; Schmidhuber, 2015). By layering multiple "hidden" processing stages between inputs and outputs, artificial neural networks (ANNs) form the core of deep learning, a branch of machine learning that builds hierarchical representations of features. First introduced in the 1940s and 1950s as the perceptron, **neural networks have experienced alternating waves of enthusiasm and skepticism until the recent "deep learning revolution," when the availability of abundant data, powerful hardware, and novel architectures reignited widespread research and real-world applications (LeCun et al., 2015; Schmidhuber, 2015).**

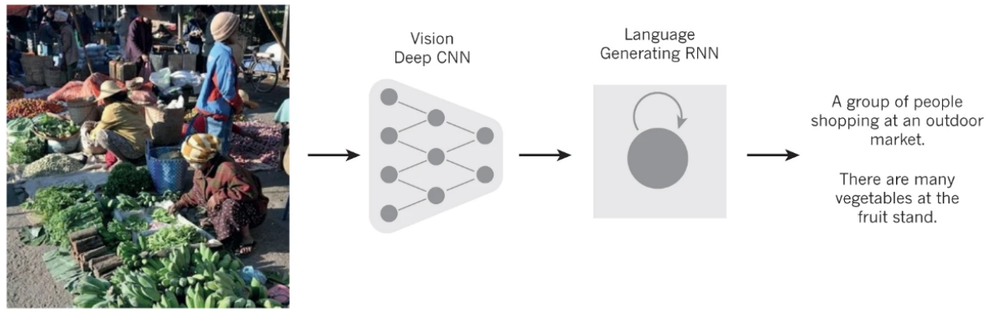


Figure 21: Applications of neural networks by Y. LeCun, Y. Bengio, & G. Hinton, 2015. https://doi.org/10.1038/nature14539

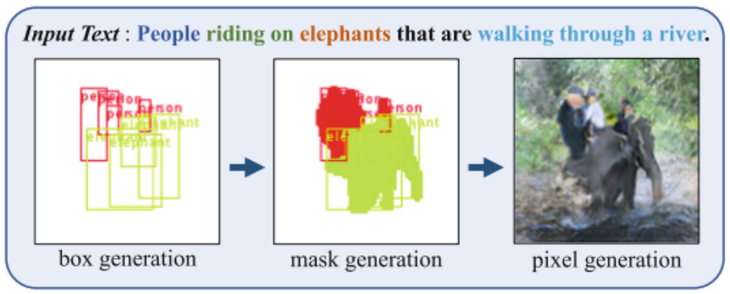


Figure 18: Command prompt image generation by Y. LeCun, Y. Bengio, & G. Hinton, 2015. https://doi.org/10.1038/nature14539

Deep neural networks function as "black boxes": inputs and outputs are visible, but the nonlinear transformations of billions of weights inside are not (Adadi & Berrada, 2018). Unlike "white box" models such as linear regression or decision trees, whose rules can be inspected, the depth and complexity of these nets make tracking any input's influence on a prediction virtually impossible (Adadi & Berrada, 2018; Montavon, Samek, & Müller, 2018). Their hierarchical features often capture abstractions that bear little resemblance to human concepts, especially in vision tasks (Buhrmester, Münch, & Arens, 2021). As models scale, emergent behaviors arise unpredictably from parameter interactions, compounding opacity (Samek et al., 2020). And because the networks' rules evolve through gradient-based training on vast datasets, they leave no human-readable record linking specific examples to particular weight changes, which complicate interpretability, accountability, and trust (Samek et al., 2020).

Why These Definitions Matter

Establishing clear, consistent definitions for circularity, R-strategies, life-cycle methodologies, digital tools, computational intelligence, neural-network characteristics, and end-of-life scenarios lays the groundwork for rigorous inquiry into how advanced computational methods can operationalize circular principles in architecture. Without shared vocabulary, practitioners and researchers risk misalignment when translating high-level sustainability goals into tangible design decisions. For instance, understanding EoL pathways is essential for embedding cradle-to-grave assessments early in the design process, rather than relegating them to post-occupancy audits. **Similarly, distinguishing between "white-box" and "black-box" computational approaches informs the selection of explainability techniques that foster stakeholder trust.** Together, these foundational concepts frame the subsequent exploration of how graph neural networks and real-time feedback interfaces can integrate life cycle thinking into the architect's earliest sketches, ensuring that circular-economy objectives drive design rather than trail behind it.

B.2 The Impact Of Decision Making Arcoss The Design Phases

Early architectural decisions—made during the conceptual and schematic phases—exert a disproportionate influence on resource flows, carbon emissions, and life-cycle performance (Dym et al., 2005; Dorst & Cross, 2001). **At these stages, project teams handle high-level abstractions of form, spatial relationships, and programmatic requirements, often under deep uncertainty. Although setting circularity goals early can establish broad targets (e.g., material reuse rates, carbon budgets), the absence of granular life-cycle data means designers struggle to translate objectives into concrete choices about geometry, material selection, and assembly methods.**

As projects advance into design development, the granularity of information increases, structural systems are defined, material specifications are selected, and detailed modeling occurs. **However, by this point, foundational form decisions are effectively locked in, constraining the range of feasible circular strategies. Without embedded feedback mechanisms that surface the life-cycle implications of early decisions, teams resort to late-stage optimizations or post-hoc retrofits that yield marginal benefits relative to the costs incurred (Eastman, Teicholz, Sacks, & Liston, 2011).**

Bridging this temporal gap requires decision-support frameworks that deliver interpretable life-cycle insights in real time, aligned with the level of abstraction at each design phase. By embedding predictive analytics directly within conceptual modeling tools, architects can iteratively explore trade-offs between spatial configurations and environmental outcomes. Such an approach democratizes sustainability decisions, enabling multidisciplinary teams to collaborate on circular solutions before the cost of change escalates.

B. Literature Review

B.2 The Impact Of Decision Making Arcoss The Design Phases

This granular understanding allows us to identify critical integration points for embedding goals in the design process and address data gaps that may impede their realization, especially regarding circularity (Jockwer et al., 2023; Passarelli, 2022; Westerholm & Franssila, 2023).

Conceptual Design

Characterized by high-level creativity and broad visions, the conceptual design phase significantly influences a project's sustainability, affecting approximately 50% of its Life Cycle Assessment (LCA) and 40% of its End-of-Life (EOL) recoverability (Ahn & Dodoo, 2022; Passarelli, 2022). During this stage, low-fidelity sketches and preliminary material selections are common. Opting for low-carbon materials, such as timber, can drastically reduce embodied carbon. Additionally, incorporating modular grids, prefabrication, and reversible connections enhances the potential for material reuse (Tsavdaridis et al., 2023). However, challenges include limited material data and a lack of integration with sustainability assessment tools (Riggio & Hasani, 2025).

Architectural Design

In the architectural design phase, broad visions are translated into more defined layouts and structural frameworks, providing a crucial opportunity to embed circularity goals (Schuster & Geier, 2022). This stage involves moderate-fidelity designs with clearer spatial configurations, preliminary structural elements, and initial material specifications. Detailed Bill of Materials (BOM) data enables the exploration of lower-impact material alternatives. Additionally, refining connections for easier disassembly through Design for Disassembly (DfD) and validating circular strategies using parametric design and preliminary Life Cycle Assessment (LCA) facilitate iterative assessments. Nonetheless, data fragmentation and interoperability issues between different software platforms pose significant challenges.

Technical Design

The technical design phase translates the architectural vision into precise construction documents, solidifying circularity through detailed specifications and fabrication plans (Riggio & Hasani, 2025). This stage involves high-fidelity drawings, models, and comprehensive specifications. Material optimization is achieved through precise selection procedures , while ensuring the constructability of DfD by detailing connections for efficient disassembly. Simulations are used to guarantee long-term performance and circularity within construction constraints (Schuster & Geier, 2022). Challenges in this phase include maintaining data coherence across specialized software and balancing competing priorities with circularity goals (Finch et al., 2021).

Understanding the nuances of each design phase allows us to create targeted circularity strategies, making circular timber construction achievable (Westerholm & Franssila, 2023).

Conceptual Design

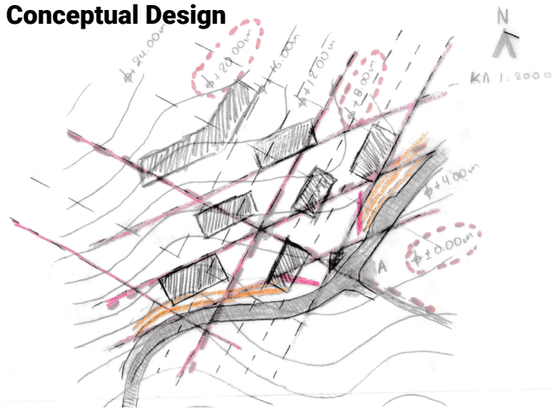


Figure 19: Concept sketch of a master plan
Source: Author,2022

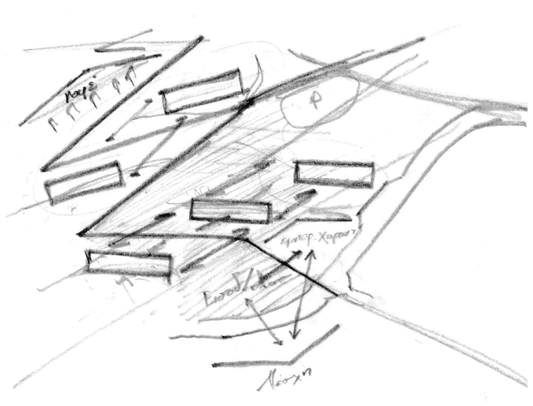


Figure 20: Master Plan Sketch
Source: Author, 2022

Client Feedback

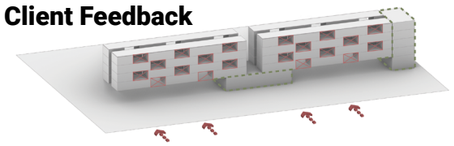


Figure 21: Mass Formation
Source: Author, 2022

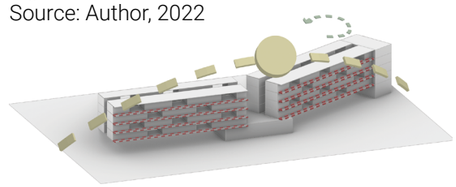


Figure 22: Circulation scenario sketch
Source: Author, 2022

Architectural Design

Figure 23: Architectural Drawing: South View
Source: Author, 2022

Client Feedback

Technical Design

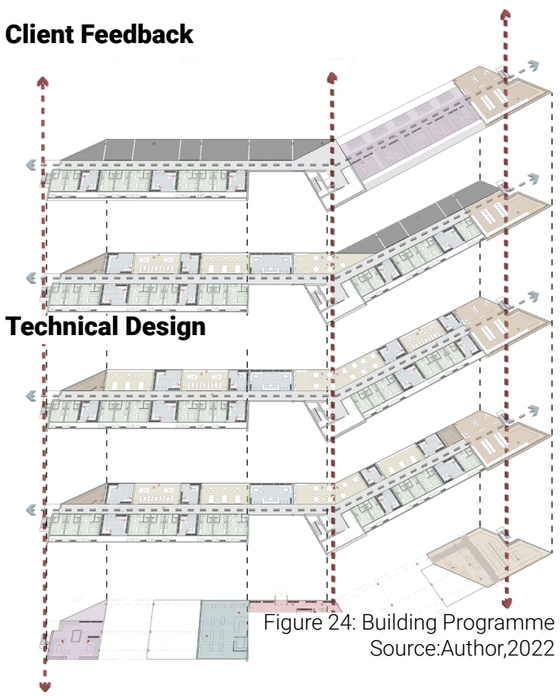


Figure 24: Building Programme
Source:Author,2022

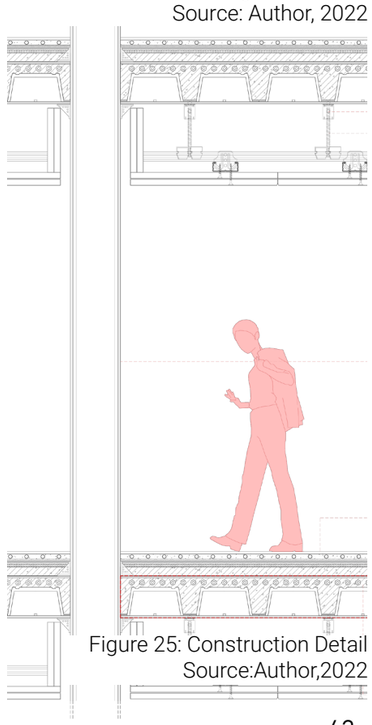


Figure 25: Construction Detail
Source:Author,2022

B. Literature Review
B.2 The Impact Of Decision Making Arcoss The Design Phases

A more targeted look at Conceptual Design

Conceptual design fixes the trajectory of environmental performance while drawings are still loose and choices remain fluid. Three adjustable levers sit at the designer's fingertips.

First, geometry establishes the quantity and distribution of built mass. Compact forms with favorable surface to area volume ratios trim envelope area, moderate heating and cooling loads, and shrink the stock of cladding that will later require recovery (Eslami, 2024).

Second, the material palette determines the ecological profile of every kilogram installed. Selecting products with documented low extraction impacts, robust reuse pathways, and transparent supply chains can lower embodied carbon by roughly thirty percent when compared with conventional assemblies (Acar Tschunko, 2022). Early clarity on sourcing also reduces procurement risk and supports reliable circular accounting.

Third, spatial layout choreographs circulation, daylight, and building services. Clear modular grids, repetitive spans, and uncluttered cores raise the rate of material utilization, streamline operational efficiency, and simplify future adaptation. When rooms align with modular dimensions, off-cut waste falls and the effort needed to repurpose spaces in later life cycles shrinks (Eslami, 2024).

Alongside these adjustable levers stand constants that frame every option. Site orientation, ground conditions, and access set physical boundaries; climate informs passive strategies and envelope thresholds; codes anchor safety, accessibility, and minimum performance. Treating these factors as partners rather than constraints allows variable decisions to stretch toward more ambitious resource goals.

The dialogue between variables and constants is measured through a family of impacts already visible in outline at this stage. High material utilization signals economical use of stock lengths and sheet goods. Favorable surface to area volume ratios predict lower operational demand. Reduced embodied carbon confirms alignment with planetary limits. Strong operational efficiency points to lean circulation and services. Healthy resource availability shows reliance on abundant or rapidly renewable inputs. Low waste potential indicates that assemblies can be taken apart and cycled back into value chains rather than landfilled (Acar Tschunko, 2022; Eslami, 2024).

Because later phases mainly refine what is established here, setting explicit targets for these metrics now ensures that every subsequent detail amplifies rather than corrects the original circular ambition.

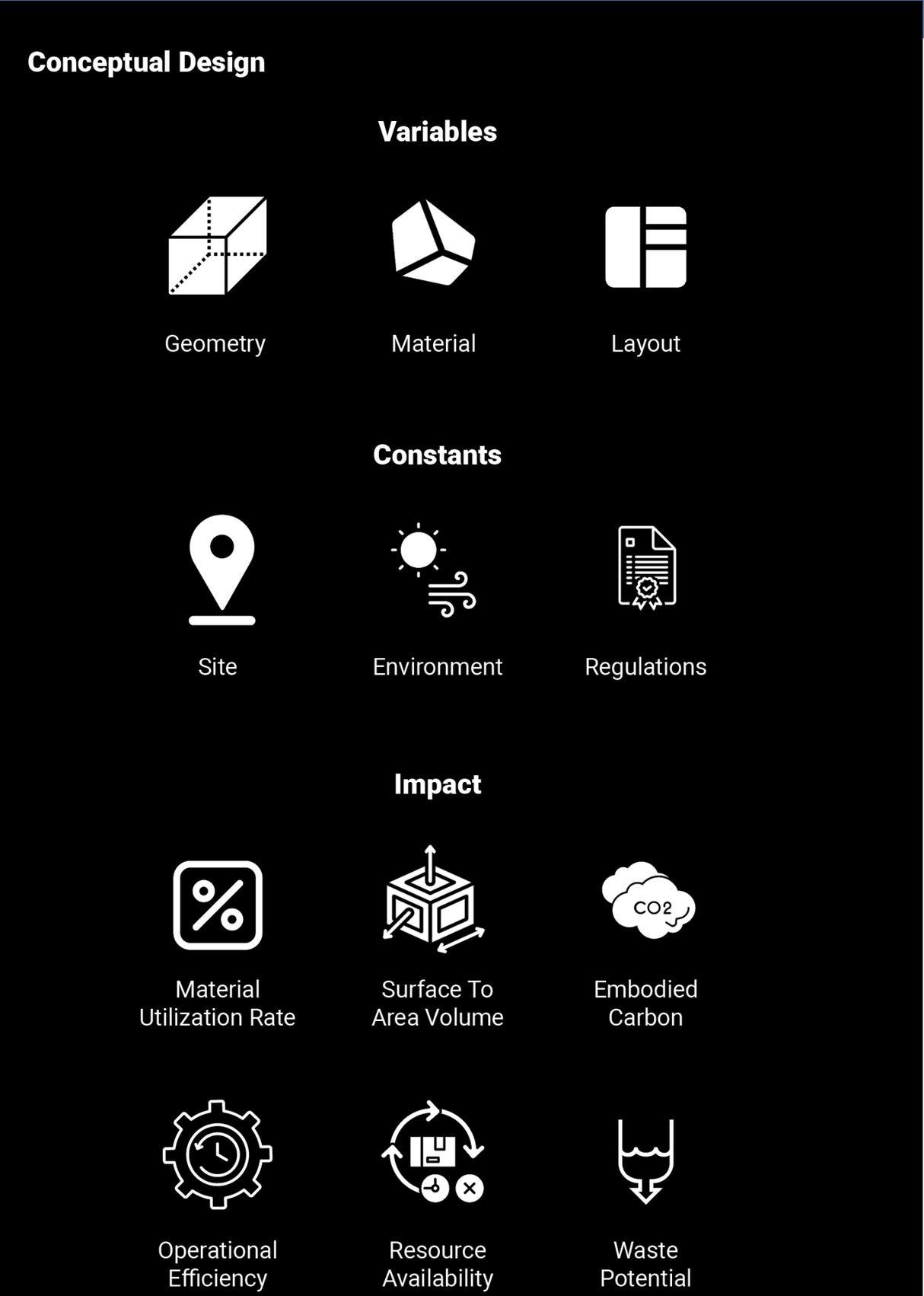


Figure 26: Conceptual Design
Source: Author, 2025

B. Literature Review
B.2 The Impact Of Decision Making Arcoss The Design Phases

A more targeted look at Architectural Design

The architectural design phase converts early vision into coordinated strategies through careful work on organisation, structural system, and detailed material choice. Variable elements such as spatial organisation, structural layout, and the evolving bill of materials interact with fixed boundaries that arise from client briefs, energy targets, and initial building physics studies. Decisions at this point establish the foundations for long term performance and circularity potential, influencing adaptability, feedback iteration, and eventual conflict resolution among stakeholders (Jockwer et al., 2023; Schuster & Geier, 2022).

Reliable data drive these choices. A refined bill of materials makes it possible to locate high impact items and substitute lower impact options while keeping functional demands intact (Ahn et al., 2022). Clear information on joints and assemblies supports design for disassembly so that future upgrades or decommissions can occur with little or no waste (Li et al., 2024). When this information circulates smoothly across disciplines, collaboration strengthens and design refinements stay both sustainable and constructible (Khan et al., 2024).

Circularity is further embedded by aligning façades, primary structure, and interior systems with modular grids and standardised components that allow repeatable, reversible connections. Examples include bolted frame systems and panelised envelope solutions that can be lifted out of a building for reuse rather than demolition (Jockwer et al., 2023; Bowyer et al., 2023; Passarelli, 2022). Life cycle assessment models with high fidelity and parametric tools give instant feedback, allowing teams to test how each design move shifts embodied carbon and operational efficiency before locking in commitments (Finch et al., 2021).

Yet the phase often suffers from data fragmentation. Structural analysis platforms, thermal simulation tools, and environmental assessment software rarely share a common schema, which complicates consistent sustainability reporting (Khan et al., 2024). Integrated workflows and truly interoperable tool sets are therefore essential to maintain accurate metrics and avoid conflicting recommendations (Kanters, 2018).

To maximise project sustainability, performance indicators must be embedded directly in the design environment so that architects receive real time guidance while selecting assemblies or adjusting geometry (Schuster & Geier, 2022). A life cycle mindset that spans sourcing, construction, use, and end of life ensures that each material choice supports long term value recovery (Hasani & Riggio, 2025). Ongoing collaboration among architectural, structural, and environmental specialists aligns objectives and reduces later conflict, while lessons drawn from completed projects feed an evolving knowledge base that continuously strengthens circular practice (Jockwer et al., 2020; Bowyer et al., 2023).

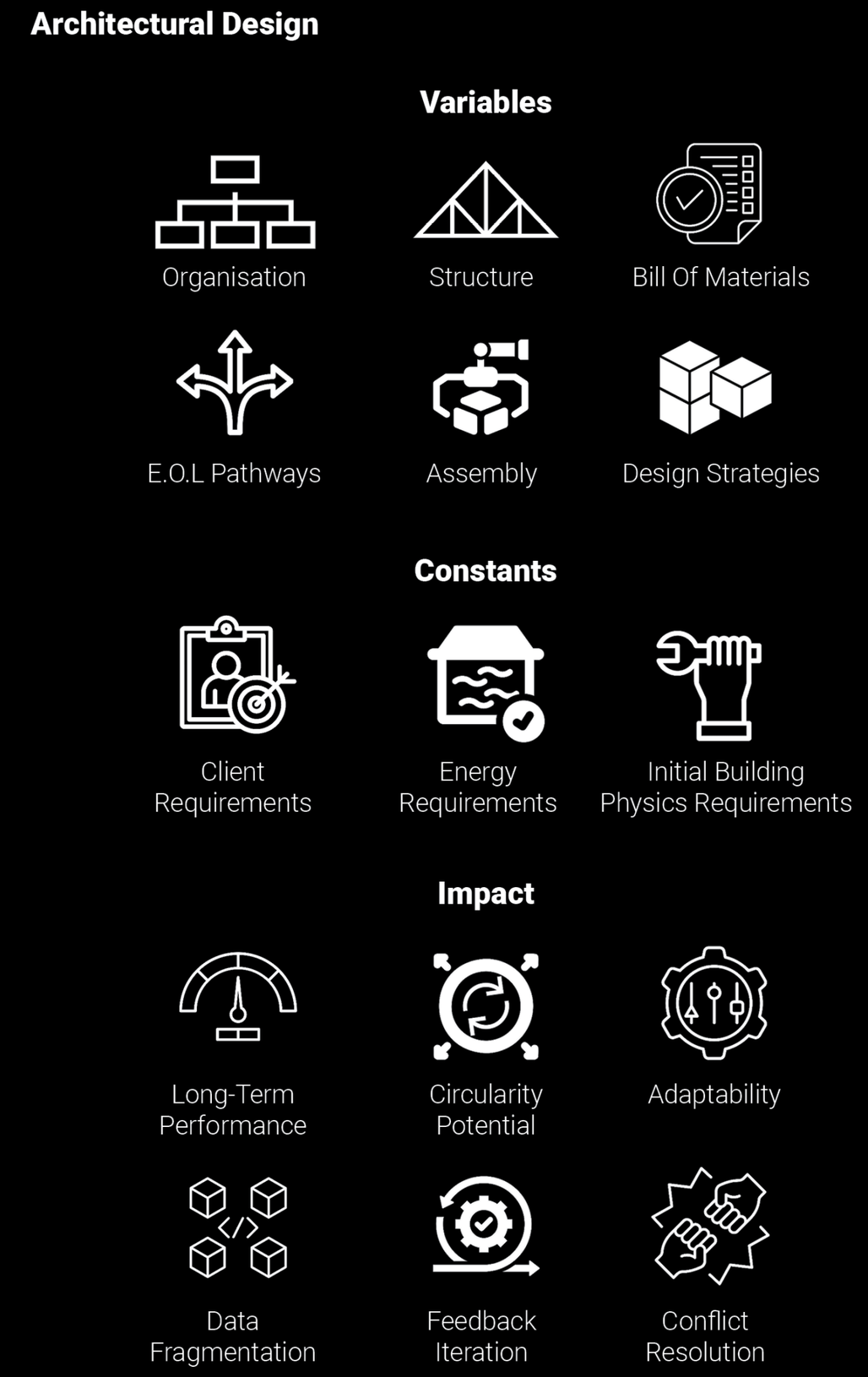


Figure 27: Architectural Design
Source: Author, 2025

B. Literature Review
B.2 The Impact Of Decision Making Arcoss The Design Phases

A more targeted look at Technical Design

Technical design converts the architectural concept into exact drawings, specifications, and digital fabrication data, making circular goals executable on site (Hasani & Riggio, 2025; Jockwer et al., 2023). Teams work simultaneously on connections, MEP routing, fabrication tolerances, component interfacing, and possible material substitutions. Each choice must stay within fixed boundaries of project budget, factory capacity, and code compliance while still driving down the embodied carbon footprint and boosting assembly efficiency.

High quality feedback is essential because structural engineers, MEP consultants, and manufacturers often rely on separate software environments. When their datasets converge in a shared BIM platform, the team can test how a new fastener type or pipe run affects adaptability and end of life reclaim potential before anything reaches the shop floor (Khan et al., 2024; Schuster & Geier, 2022).

Material strategy meets practical constraints here. A bolted frame may need custom hardware that satisfies strength requirements yet also unlocks straightforward disassembly, which can raise cost or carbon unless carefully optimised (Eslami, 2024). Advanced simulations and real time fabrication feedback explore options that limit waste and conserve resources without delaying the schedule (Bowyer et al., 2023; Kanters, 2018).

Coordinated data integration and iterative validation keep earlier circular commitments intact. By tracking maintenance cycles, retrofit potential, and occupant well-being alongside structural checks, the team translates abstract sustainability aims into measurable construction outcomes (Jockwer et al., 2023; Li et al., 2024). Performance models verify that tolerances, seals, and interfaces will deliver the promised energy gains while still permitting reuse at end of life.

Although initial performance estimates arise in conceptual design, technical detailing is where those assumptions become products with serial numbers. Continuous refinement of circular targets during architectural and technical design embeds cradle to cradle thinking in every bolt and gasket so that each revision enhances rather than dilutes the original objective set-out for the project (Ahn et al., 2022; Hasani & Riggio, 2025; Finch et al., 2021).

Figure 28: Technical Design
Source: Author, 2025

B. Literature Review

B.3 Circularity And The Absence Of Informed Decision-Making

Despite a wealth of sustainability frameworks and increasingly sophisticated computational tools, the integration of circular economy principles into architectural design remains fragmented and inconsistent. In practice, designers still rely heavily on standalone utilities such as Building Information Modeling platforms that focus on geometry and coordination (Eastman & Eastman, 2008), finite element analysis packages that optimize structural performance, and late-stage life cycle assessments that quantify environmental impacts only after schematic design is complete (Pomponi & Moncaster, 2017; Finnveden et al., 2009). **Key decisions about form, materials, and structural systems are therefore made without real-time feedback on carbon, energy, or waste flows, consigning circularity to a retrospective compliance exercise rather than allowing it to shape the design from the outset (Kirchherr, Reike, & Hekkert, 2017).**

Modern digital workflows compound this disconnect by siloing data at every step. Environmental Product Declarations are published in external repositories with minimal interoperability (Del Borghi, 2013). Structural analyses proceed in dedicated finite element software divorced from material databases. Predictive machine learning models for performance assessment are developed in isolation from the BIM environment (Adadi & Berrada, 2018; Doshi-Velez & Kim, 2017). **By the time sustainability metrics finally enter the process, often during detailed design or construction documentation, the opportunity to explore low-impact alternatives at the conceptual stage has vanished and the cost of substantive redesign becomes prohibitive.** This temporal and informational disconnect stifles innovation and limits meaningful reductions in embodied carbon and resource throughput (Ghisellini, Cialani, & Ulgiati, 2016).

Bridging this gap requires a paradigm shift from isolated tools to an interconnected data-driven workflow that embeds life-cycle insights from the very outset of design. The European Commission Joint Research Centre's model for building LCA provides a framework to integrate EPD-derived metrics into conceptual massing studies (European Commission JRC, 2018a). Explainable artificial intelligence techniques such as transparent neural-network interpretability methods can be coupled with BIM to deliver real-time insights on embodied carbon, energy use, and material throughput (Montavon, Samek, & Müller, 2018; Molnar, 2022). Unifying geometric modeling, structural optimization, and environmental assessment within a single explainable framework allows circularity to become a driving constraint so that every sketch and every structural decision can be iterated against quantified sustainability targets, transforming the circular economy from an afterthought into a foundational design principle.

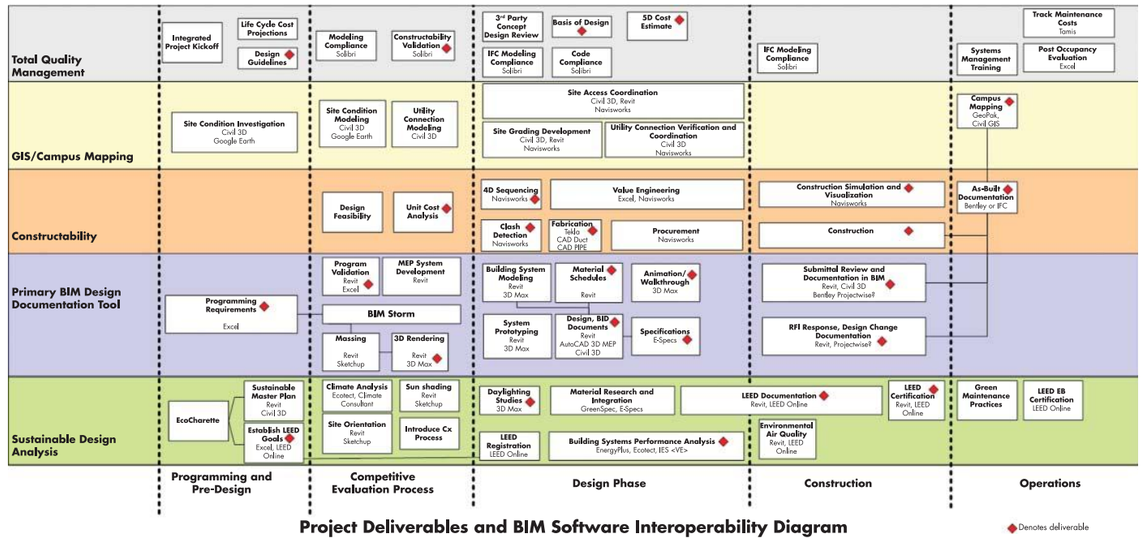


Figure 29: List of isolated project deliverables in a BIM workflow, an increased numbed and a project processed in silos. Source: Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). BIM handbook: A guide to Building Information Modeling for owners, designers, engineers, contractors, and facility managers* (3rd ed.)

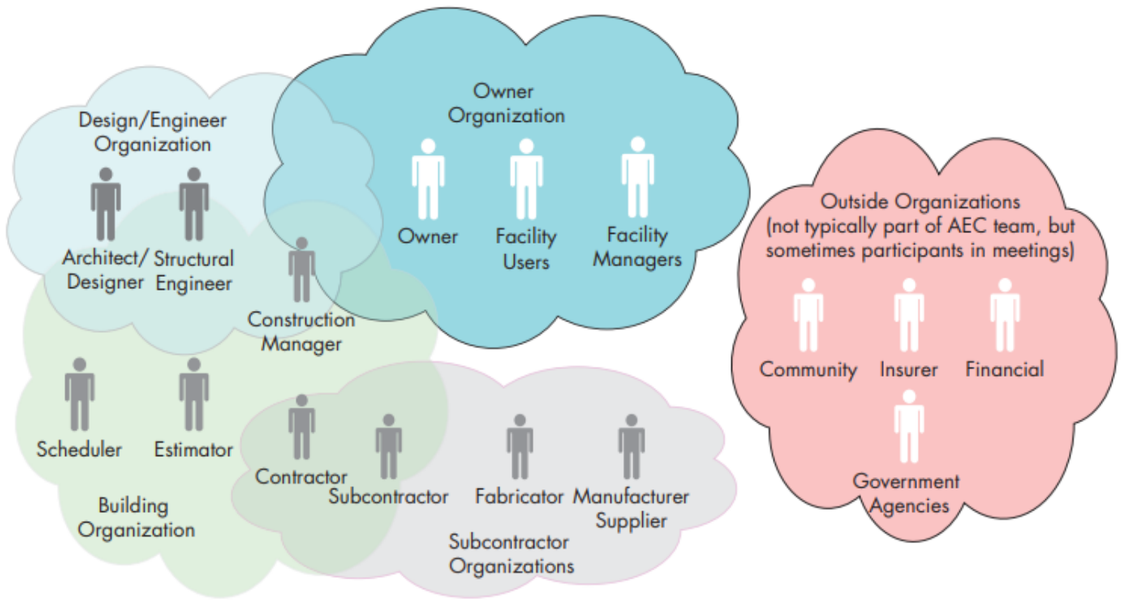


Figure 30: Conceptual diagram representing an AEC project team and the typical organizational boundaries. Source: Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). BIM handbook: A guide to Building Information Modeling for owners, designers, engineers, contractors, and facility managers* (3rd ed.)

B. Literature Review
B.4 The Matching Of Circular Strategies And Digital Tools

The digital transformation of architectural practice has elevated circular design from a detached audit to an immersive data driven ecosystem. Advanced Building Information Modeling platforms now integrate material passports and preliminary life cycle modules during conceptual massing (Eastman 2008; European Commission Joint Research Centre, 2018b). Parametric design tools and generative algorithms enable architects to explore design variants against embedded sustainability metrics (Molnar, 2022; Doshi-Velez & Kim, 2017). Immersive environments such as virtual reality workspaces and digital twins allow stakeholders to visualize and interact with circular performance insights in real time, fostering collaborative decision making at the earliest stages of design (Getuli et al., 2020; Dym et al., 2005).

Critical data on sourcing, reuse pathways and end of life impacts often remain scattered across disparate repositories. **Environmental Product Declarations sit in isolated databases with limited interoperability (Del Borghi, 2013; Gervasio et al., 2018).** Structural analyses occur in standalone finite element applications disconnected from these material libraries and from the BIM environment (Ma et al., 2024). Machine learning models for performance assessment are frequently developed in research silos without standardized data exchange protocols (Wu et al., 2021; Özerol & Arslan Selçuk, 2023). As a result key circular criteria are unavailable during initial sketches and early massing studies and must often be introduced later at significant cost (Ghisellini, Cialani, & Ulgiati, 2016).

Bridging this gap requires an integrated workflow that merges conceptual models with comprehensive life cycle databases and explainable predictive analytics from the very beginning of design.

The European Commission Joint Research Centre LCA framework can be embedded directly into BIM authoring environments to deliver real time feedback on embodied carbon, material throughput and energy use starting with the first sketches (European Commission Joint Research Centre, 2018a). Explainable artificial intelligence methods from transparent neural network interpretability to surrogate modeling can drive generative design loops that optimize form, material and structural decisions against quantified sustainability targets (Adadi & Berrada, 2018; Montavon, Samek, & Müller, 2018). Unifying geometric modeling, structural optimization and environmental assessment within a single data driven ecosystem transforms circular economy principles into a design driver rather than a compliance checkpoint.

Figure 31: BIM model of the workspace configuration for the installation activity of a Cross-Laminated Timber wall panel: top, photo of the activity taken on site (30/07/2018); center, workspace model axonometric view; bottom, workspace model plan view (dimensions in [cm]).
Source: Getuli, V., Capone, P., Bruttini, A., & Isaac, S. (2020). BIM-based immersive virtual reality for construction workspace planning: A safety-oriented approach,Automation in Construction, 114, Article 103160. https://doi.org/10.1016/j.autcon.2020.103160

B. Literature Review
B.5 Building Module | Prefabrication In Modern Practice

The research's computational agenda seeks to give designers real-time, explainable life-cycle feedback but that vision can only be realized if the construction industry adopts truly industrialized workflows. Modular prefabrication provides the necessary physical foundation: by fabricating LVL modules in a factory's tightly controlled environment, we ensure consistent material quality, precise joinery and minimal waste. These modules then travel as finished units complete with all openings, connections and annotations enabling rapid on-site assembly that dramatically shortens project schedules. This end-to-end predictability not only drives down embodied-carbon and transportation emissions through optimized batch production and logistics, but also aligns sustainability targets with reliable cost estimates, streamlined labor requirements, and superior build quality.

Mass timber systems illustrate how modularity and sustainability reinforce each other. Their high strength to weight ratios allow large, repeatable panel formats that travel well, tolerate computer numerical control machining, and carry much lower embodied carbon than concrete or steel (Abed et al., 2022). A cradle to gate assessment confirms significant global warming potential savings for laminated veneer lumber compared with functional alternatives (Bergman & Alanya Rosenbaum, 2017). Factory production also minimises off cuts and simplifies recycling loops, so timber's stored carbon remains in use rather than entering waste streams.

Industrialised construction is above all an information management task: part data captured at concept stage must flow without distortion into computer aided manufacturing files, logistics plans, and site sequencing. Building Information Modelling first highlighted this continuity requirement (Eastman & Eastman, 2008). Recent work shows that parametric models can stream environmental metadata directly into iterative design moves, allowing teams to adjust module geometry while monitoring embodied carbon in real time (Ma et al., 2024).

Prefabrication shortens schedules and tightens tolerances, yet it also demands revised safety protocols. Immersive virtual reality linked to Building Information Modelling lets teams rehearse crane paths and identify collision hazards before modules arrive, something possible only when unit dimensions and connection details are frozen early (Getuli et al., 2020).

Modular units enable reuse and refurbish strategies because whole volumes can be removed intact and redeployed. Case studies record higher material circularity scores for timber modules than for masonry, mainly because reversible mechanical connections permit clean disassembly (Leising et al., 2018). European guidance strengthens this promise by calling for dimensional coordination and digital passports that follow each unit through successive life cycles (European Commission Joint Research Centre, 2018b).

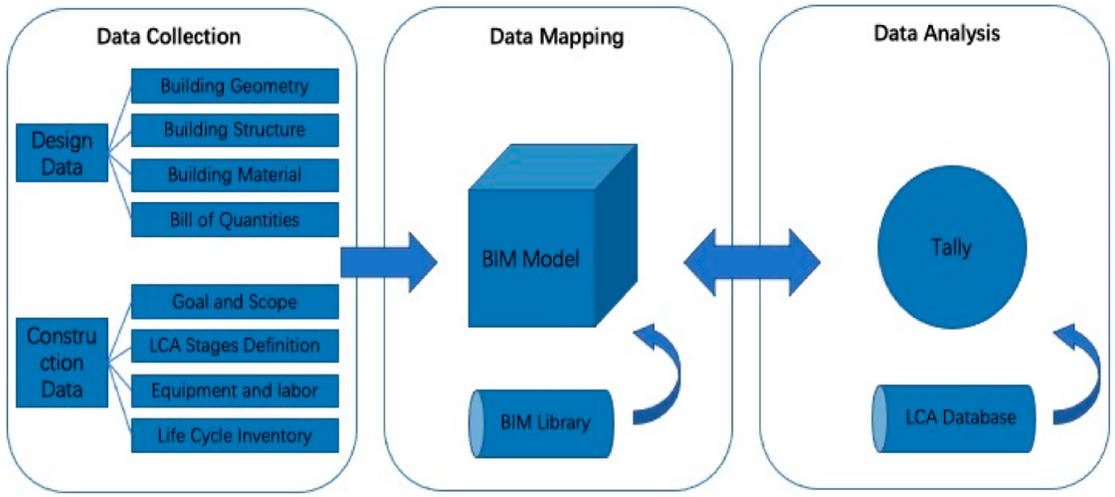


Figure 32: BIM based LCA calculation
Source: Ma, L., Azari, R., & Elmeiri, M. (2024). A Building Information Modeling-based life cycle assessment of the embodied carbon and environmental impacts of high-rise building structures: A case study https://doi.org/10.3390/su16020569

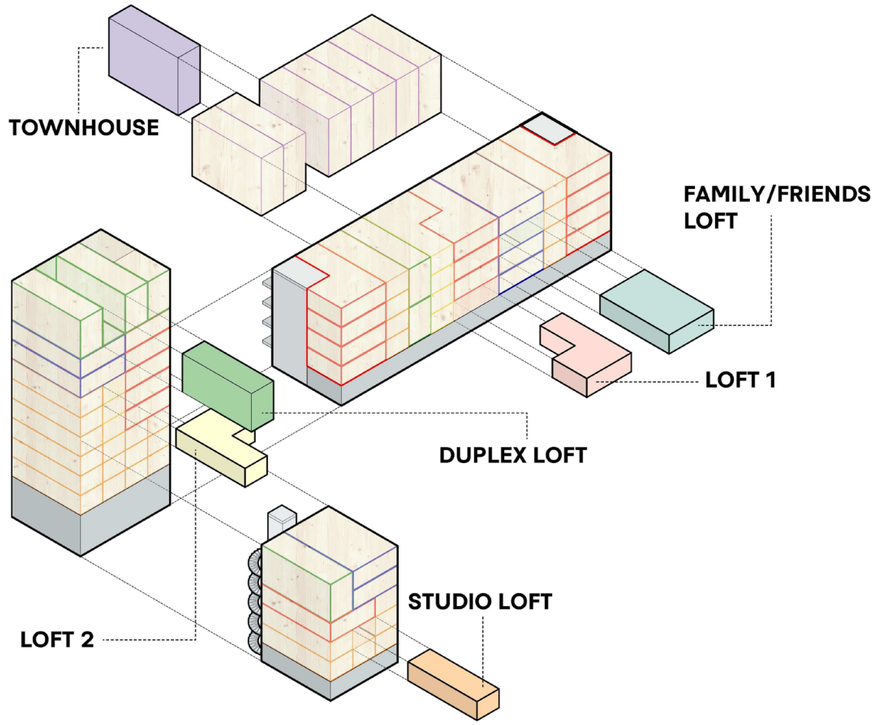


Figure 33: Mass programming in modular design.
Source: https://marckoehler.com/story/superwood-module-timber-housing/

B. Literature Review
B.5 Building Module | Prefabrication In Modern Practice

What is a building module in this research?

By collaborating with Sustainer for this project and by using their definition of a building module, it is a transportable cell built from laminated veneer lumber panels that are fully braced for structural rigidity. The floor cassette uses box beams topped with a structural panel. Walls consist of twin veneer skins stitched by plywood ribs, routed in the factory for service runs. A services spine pre-installs ducts, water pipes, and electrical conduits so that once a module reaches the site the trades connect with simple plug and play fittings. Four corner plates and intermediate knife plates align adjoining modules within one millimeter tolerance and transfer both gravity and lateral loads into the floor cassette. Inside, most finishes such as gypsum board, acoustic insulation, windows, and doors are factory mounted, leaving mainly inter-module sealing and façade work for the site crew.

Two ways to view the same module while using the Sustainer digital toolkit in Rhinoceros 3D

Basic level
At the earliest design moment, the module is nothing more than a three dimensional box that expresses footprint, height, and zoning intent. Designers can stretch the box in fixed increments and subtract openings as simple surface rectangles.

Construction level
When the designer selects the construction command, the same box becomes a fully detailed unit. The toolkit populates its shell with all the necessary components for manufacturing.Openings turn into framed inserts with lintels and sill plates. The output is a manufacturing ready IFC.

This dual reading, conceptual box for speed and fabrication rich geometry for execution, ensures that early life cycle estimates rest on components that are both buildable and certifiable, thus keeping digital analysis and physical delivery in lockstep.

Factory productivity depends on large production runs, yet bespoke architecture often resists standardisation, requiring design research to reconcile unique spatial programmes with repeatable modules. Environmental data for hybrid subassemblies such as service cores that merge fibre cement panels with mechanical, electrical, and plumbing lines also remain incomplete, hindering full optimisation across interfaces.

Modular prefabrication therefore elevates off site construction from a cost saving tactic to a primary channel for measurable carbon abatement, rapid delivery, and circular material stewardship. Its promise becomes tangible when digital continuity, dependable environmental data, and reversible assemblies converge, conditions that the research's explainable artificial intelligence framework is designed to exploit.

Figure 34: A building module.
Source: https://www.blumer-lehmann.com/en/implement-construction-projects/construction/modular-construction.html

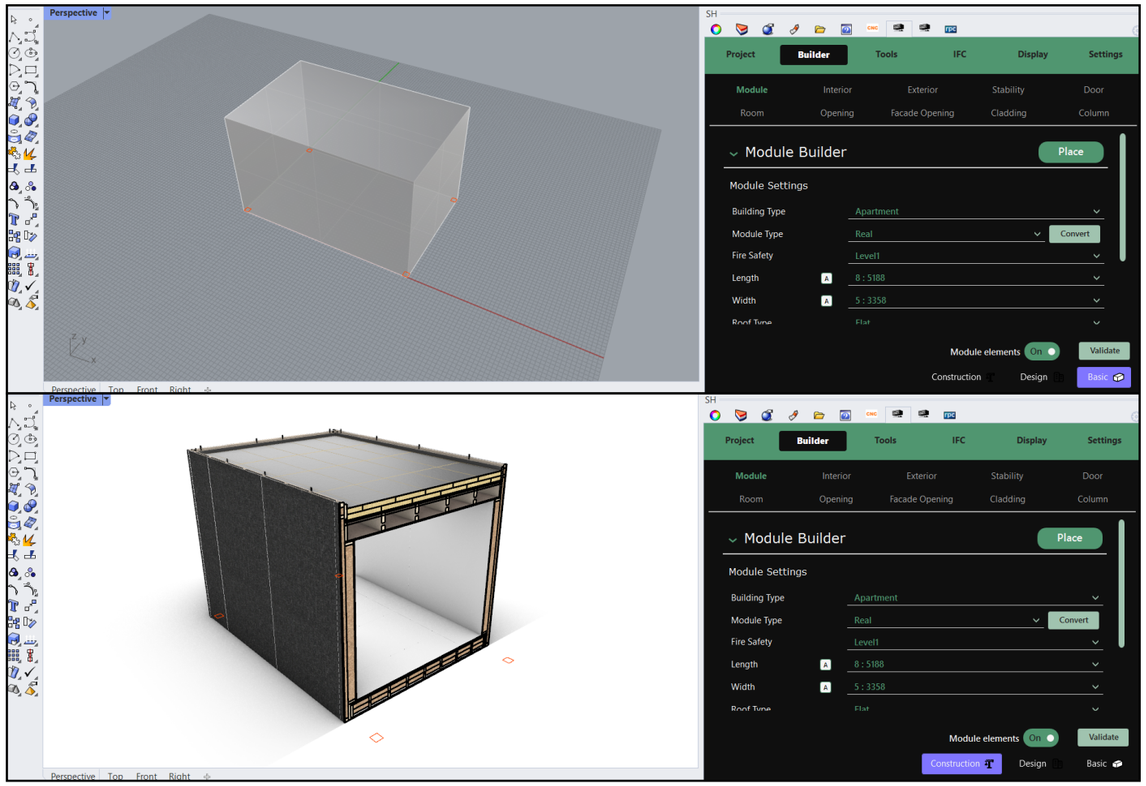


Figure 35:The Sustainer building module / Basic & Construction Level
Source: Author, 2025

B. Literature Review
B.6 Laminated Veneer Lumber

Laminated Veneer Lumber (LVL) is the cornerstone material that allows the research to connect its three strands: (i) early stage life cycle assessment, (ii) explainable graph-based prediction, and (iii) factory made modular construction. Choosing one flagship product keeps the study narrow enough for deep data integration yet broad enough to test the project's larger claim that circular performance can inform concept sketches rather than audit finished drawings.

Manufacturing logic and material behavior
LVL is produced by rotary-peeling softwood logs into thin veneers, drying them, applying phenol-formaldehyde or melamine-urea-formaldehyde adhesive, and then arranging the sheets so that grain directions run parallel before hot pressing (Bergman & Alanya-Rosenbaum, 2017). The process yields uniform stiffness and strength values that exceed those of sawn timber of comparable grade, which means designers can rely on small cross-sections and longer spans without resorting to steel ties or concrete ribs. Controlled factory pressing also eliminates the natural defects that complicate strength grading in solid wood. These characteristics underpin the dimensional accuracy demanded by the modular strategies described in Section B.5, ensuring that factory-cut panels align within the one-millimeter tolerances required for rapid on-site assembly.

Environmental performance across the life cycle
From a cradle-to-gate perspective LVL embodies considerably less fossil carbon than concrete or steel structural systems of equivalent capacity (Abed et al., 2022). The US life cycle inventory assembled by Bergman and Alanya-Rosenbaum (2017) reports global-warming potential figures below 300 kg CO₂-eq per cubic meter, including all upstream processes. When biogenic carbon storage is credited under EN 15804 scenarios, net values can be near or even below zero during the product phase, a leverage point explicitly targeted in the research questions that ask how early modelling can reward low-carbon choices. European EPDs used in the study expand this inventory by adding transport, installation, use, and end-of-life stages, enabling the computational pipeline to simulate cradle-to-grave or cradle-to-grave-and-beyond pathways in real time.

Compatibility with digital-first design workflows
LVL arrives from the press in large, predictable panel sizes that accept computer-numerical-control routing without splintering. This machinability aligns with the parametric toolkit discussed in Section A.8, where module geometries and opening patterns are manipulated live in Rhinoceros 3D. Because veneer sheets are thin, waste off-cuts can be shredded and re-introduced into composite panels, supporting the circular ambition outlined in the introduction.

Knowledge gaps addressed by the research
Despite these advantages, three critical uncertainties remain and frame the project objectives:

1.Data fragmentation
Manufacturers publish EPDs with varying system boundaries and end-of-life assumptions. The research reconciles these discrepancies inside a unified lookup table that feeds the graph model, answering Objective 1's call for an integrated LCA framework.

2.Predictive transparency
Design teams need to know why moving a window or extending a room changes embodied carbon. By training an explainable GNN on LVL-specific assemblies, the study responds to Research Question 3, which seeks component-level insight rather than black-box scores (Doshi-Velez & Kim, 2017).

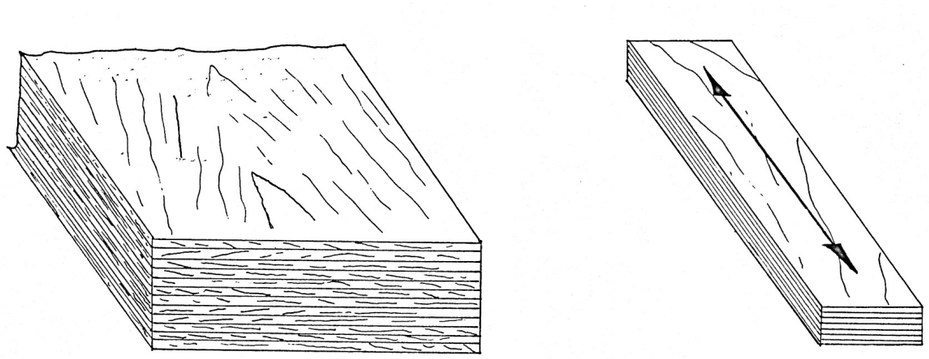


Figure 36: Laminated Veneer Lumber
Source: https://www.storaenso.com/en/products/

B. Literature Review
B.7 Life Cycle Analysis And Material Usage

Early sections of this review explained why circular design only becomes actionable when quantitative evidence is available during concept development. Life-Cycle Assessment (LCA) supplies that evidence by tracing environmental burdens and benefits from raw extraction to final disposal. Section B.7 explains how the research secures trustworthy data streams, which indicators it carries forward into the predictive model, and how end-of-life (EoL) choices close the feedback loop between design intent and circular performance. The content answers Research Objective 1, which calls for an integrated LCA framework, and prepares the ground for the explainable graph model described later.

B.7.1 Certified Means of Evaluation

International core standards
ISO 14040 and 14044 give the generic rules for goal definition, inventory analysis, impact assessment, and interpretation. In Europe these are specialised for construction through EN 15804 for products and EN 15978 for whole buildings. The European Commission's model for building LCA translates the two standards into a practical workflow and is the primary reference database in the plug-in (European Commission Joint Research Centre, 2018a).

Single score national schemes
Several member states condense the multi-category output of EN 15978 into one value that regulators can set as a limit:

-In the Netherlands the Milieuprestatie Gebouwen method aggregates nineteen midpoint categories by applying ReCiPe shadow prices and reports the result in euro per square metre of gross floor area per year. Recent research shows how this score will tighten by 2030, making it a direct design constraint (Szalay, 2024).

-The same weighting set is reused at product scale as the Milieukosten-indicator. Contractors rely on this value to obtain fictitious discounts in Dutch public tenders. Because MKI is already expressed in currency it can be added to ordinary cost plans without conversion, a feature that supports the project's aim to couple carbon and finance in one dashboard.

Macro-objective frameworks
The EU Level(s) toolbox extends EN 15978 with six headline objectives that cover life-cycle carbon, resource efficiency, water use, health, resilience, and cost. Its Indicator 1.2 requires global-warming potential for building stages A1 to C4 and aligns with impending Energy Performance of Buildings Directive updates, while Indicator 2.2 tracks the share of reused and renewable content and thus speaks directly to the modular timber strategy described in earlier sections (Ferrari et al., 2021).

Belgium's TOTEM platform and Germany's ÖKOBAU.dat library follow the same EN 15804 logic but supply pre-configured assemblies and benchmark ranges that can be compared with project output during design development (Meex et al., 2018). Although not compulsory in the Netherlands, the research keeps the underlying data keys in its lookup table so that cross-border projects can switch jurisdiction without breaking the parametric workflow.

Using multiple schemes might appear redundant, yet each serves a distinct purpose. ISO based modules secure international comparability, single-score schemes offer a permit oriented compliance lever, and Level(s) provides a bridge to future European regulation. Maintaining them side by side avoids the need for later data translation and therefore fulfils the transparency requirement set out in the project questions.

B. Literature Review
B.7.2 Necessary Metrics And Research Contribution To LCA

Before listing the indicator set, it is useful to introduce Level(s), the European Commission's voluntary framework for reporting whole-life performance of buildings. Level(s) translates the broad climate and circular-economy goals of EU policy into a common indicator suite that complements national methods such as the Dutch MPG / MKI, while remaining fully compatible with EN 15804 and EN 15978 (Ferrari et al., 2021). In practice, adopting Level(s) does not require additional calculations; it simply repackages the same EN-based data so results remain intelligible across borders.

Accordingly, every Environmental Product Declaration (EPD) employed in the study must deliver one compact yet comprehensive dataset that aligns, without translation, with EN 15804 module logic, the Dutch MPG / MKI single-score method, and the EU's Level(s) reporting framework.

At the core of this dataset is global-warming potential (GWP) expressed in kilograms of CO_2-equivalent for each life-cycle stage. GWP is non-negotiable: it is mandated by EN 15978, counted verbatim in Level(s) Indicator 1.2, and carries the heaviest weight in the ReCiPe shadow-price vector that produces the Dutch monetary score (Huijbregts et al., 2017). Coupled to GWP is biogenic carbon content, recorded as a stored-carbon credit in the product stage; omitting it would render any comparison of timber and mineral systems meaningless.

Level(s) also requires evidence of material efficiency. Two descriptors therefore travel with every product record: renewable versus non-renewable primary-energy demand and the share of reusable or recyclable content (out of scope of the research). Retaining them in physical units allows the same numbers to be re-weighted into euros for MPG / MKI compliance, keeping the monetary roll-up transparent rather than opaque.

Material mass anchors transport calculations in EN 15804 module A4 and normalises impact for functional-unit reporting, so a single kilogram count serves three regulatory dialogues simultaneously. Finally, each EPD must specify at least one end-of-life profile—incineration with energy recovery, closed-loop recycling, reuse, or landfill—mapped to stages C and D of EN 15804. If a declaration omits these data, the gap is flagged and filled with a conservative generic from a national database such as ÖKOBAU.dat, preserving traceability.

Insisting on this lean but complete indicator spine means one calculation can populate Dutch permit tables, satisfy Level(s) templates, and remain fully traceable for peer review. Rather than keeping the certified schemes of B.7.1 in separate silos, the study fuses them into a common metric language that supports both innovative design moves and regulatory due diligence.

B. Literature Review
B.7.3 End Of Life Scenarios

Circular design reaches its full value only when the designer can visualise what happens to a material after its first service life. End-of-life outcomes therefore complete the feedback loop started in the concept phase and lock the research to EN 15804 stages C and D. Rather than treating disposal as a fixed assumption, the study models four distinct pathways that differ in both physical handling and accounting treatment. Designers can move between them while they sketch, so each option remains an active design parameter rather than a post-hoc footnote.

Direct reuse of whole modules keeps the LVL panels intact and transfers their residual environmental debt to a second building. Because no re-manufacturing energy is needed, stage C emissions are limited to crane operations and transport, while stage D carries a negative credit that reflects the raw material avoided in the next project (Pomponi & Moncaster, 2017). This route best rewards the reversible connections adopted in the modular strategy and maximises the material circularity score required by Level(s) Indicator 2.2.

Closed-loop recycling chips panels and presses them into new composite boards. Mechanical comminution and pressing add energy demand in stage C, yet stage D earns a credit equal to the upstream impact of the virgin LVL it displaces (Ghisellini et al., 2016). The Dutch MKI method values this credit in euros, revealing whether design choices that complicate disassembly still make financial sense when recycled content is monetised.

Incineration with energy recovery is the default embedded in most EPDs. Biomass combustion releases biogenic carbon but produces process heat that replaces fossil fuel in a district-energy grid. EN 15804 allows this avoided burden to appear as a negative in stage D; however, the net benefit depends on regional energy mixes and can turn positive or negative as electricity grids decarbonise (Bergman & Alanya-Rosenbaum, 2017). Including this pathway maintains comparability with legacy datasets and underlines the time sensitivity of carbon-offset claims.

Landfill remains the regulatory fall-back where reuse or recycling infrastructure is absent. Anaerobic decomposition converts a fraction of stored carbon to methane, which carries a global-warming potential twenty-eight times higher than carbon dioxide over a one-hundred-year horizon. Recent EU guidance proposes default decay factors so that landfilling timber no longer appears climate neutral (European Commission Joint Research Centre, 2018b). Retaining this worst-case option exposes the penalty of design decisions that hinder disassembly and satisfies the transparency requirement in ISO 14044.

When a designer switches among these scenarios the underlying inventory does not change; only the flows that cross the system boundary move between the product line and the avoided-burden column. This transparent bookkeeping allows the single metric spine of GWP, primary energy, and monetary shadow price to remain intact while still reflecting the moral weight of end-of-life choices. The approach meets Research Objective 1 by embedding certified C and D stage logic inside the same indicator framework that governs earlier life-cycle phases, ensuring that carbon and cost dialogues stay aligned from ground-breaking to deconstruction.

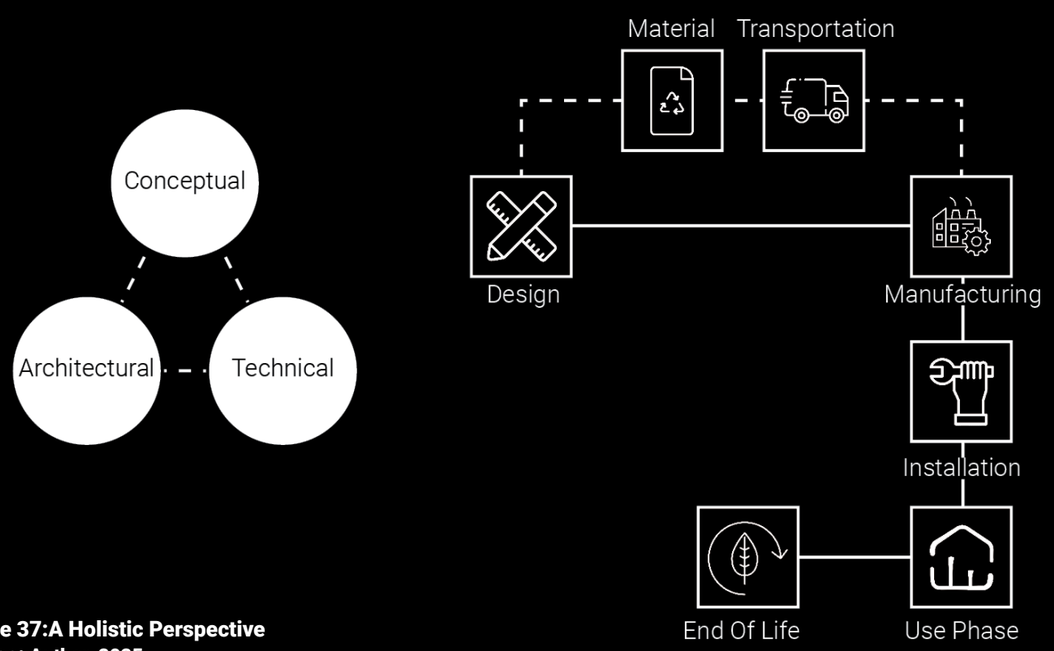


Figure 37:A Holistic Perspective
Source: Author, 2025

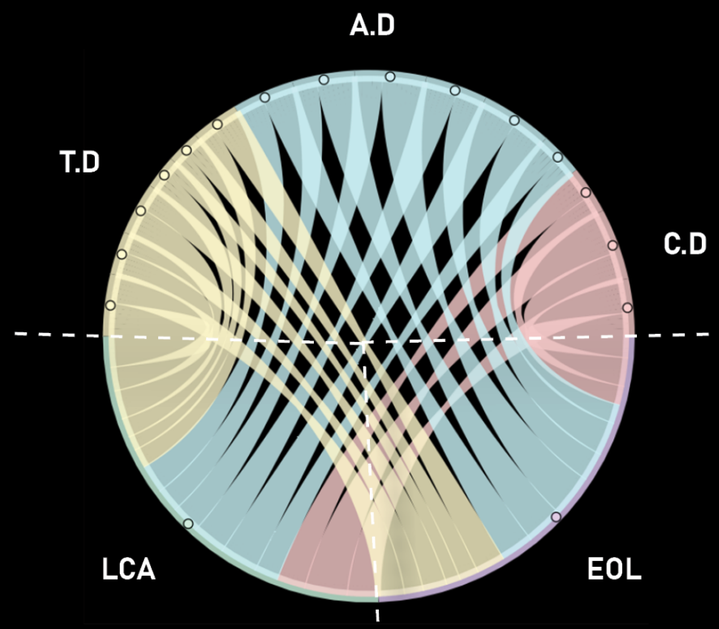


Figure 38: Connecting variables across the design phases tb EOL and LCA.
Source: Author, 2025

B. Literature Review

B.8 Ensuring Circular Outcomes

A cradle-to-cradle ambition must guide every stage of design, yet real projects rarely keep that intention intact. Time pressure, budget shifts, regulatory change, and fragmented digital tools all erode early promises. Still, progress is possible when each design phase keeps circular performance visible and negotiates compromises in full view of life-cycle evidence developed in Section B.7 and of the conceptual-stage focus defined earlier.

Conceptual Design makes structural and material choices that govern most of a building's embodied impacts. Selecting a low-carbon primary system—such as mass-timber frames or LVL modules—can halve product-stage emissions when compared with steel or concrete alternatives (Abed et al., 2022). Early design is also the moment when creative problem framing and solution search evolve together, so circular criteria must be embedded directly in that co-evolution (Dorst & Cross, 2001).

Architectural Design translates the initial concept into coordinated layouts, façade assemblies, and service routes. Disciplines often work in separate software, and full BIM-to-LCA coupling is still uncommon (Ma et al., 2024). Even so, architects can hold on to circular intent by specifying reversible joints, reserving space for retrofits, and favoring components with transparent Environmental Product Declarations (Del Borghi, 2013).

Technical Design finalizes dimensions, codes, and construction details. Major material swaps are unusual at this depth, but detailing still matters; labelled fasteners and documented disassembly paths support future recovery (European Commission Joint Research Centre, 2018b).

In practice, a fully integrated framework may be unworkable, yet partial measures and deliberate coordination in each phase can measurably reduce embodied carbon and bolster end-of-life recoverability. By remaining alert to sustainability objectives, and aligning them wherever possible despite inevitable compromises, projects can strike a more balanced path toward circular design.

B. Literature Review

B.8.1 Securing a Framework: Conceptual Or Architectural Focus?

Concept stage holds the carbon lever. Once massing and primary material are fixed, most of the future carbon cost is locked in. Capturing LCA data while the geometry is still a set of boxes—exactly where the graph model operates—makes early budgeting non-negotiable (Finnveden et al., 2009). A single secured gate at the end of concept freezes that budget and treats all later edits as controlled variations. One baseline, one audit trail.

-Conceptual checkpoint sets provisional carbon ceilings, circularity goals, and module types.
-Architectural checkpoint re-tests the same metrics once grids, façades, and services are sketched but before suppliers are locked.

B. Literature Review
B.9 The Benefits Of Real-Time Assesment

Real time environmental feedback is emerging as a practical way to keep carbon and resource goals visible while design choices are still cheap to change. Interactive platforms that couple parametric modelling with instantaneous life-cycle or energy calculations allow architects to test dozens of form and material options in minutes, greatly expanding the solution space compared with traditional, file-based LCA workflows (Mahdavian et al., 2021). Studies of dynamic life-cycle assessment driven by building information models show that projects using live metrics maintain closer alignment between early estimates and final embodied-carbon totals, typically limiting divergence to under ten percent, whereas conventional projects can drift far beyond that range (Lorenz et al., 2019). Immediate numerical and visual cues—such as colour-mapped carbon intensities on model elements—also improve interdisciplinary communication; non-specialists can identify hotspots without consulting separate reports, which shortens coordination cycles and reduces redesign effort (Pettersen et al., 2022).

When real-time tools integrate operating-energy and embodied-carbon data, designers can negotiate trade-offs on the fly, avoiding the pitfall of lowering one impact while raising another (Oti et al., 2020). Finally, rapid feedback supports policy compliance by revealing whether a scheme is on track to meet tightening carbon-budget thresholds well before statutory submissions are due, helping clients align with long-term climate commitments (Rahimian et al., 2019). In sum, the literature indicates that instant assessment shifts sustainability from a retrospective audit to a daily design habit, delivering measurable gains in accuracy, speed, and collaborative understanding.

B.9.1 Existing Industry Tools

The Importance of Feedback Mechanisms Beyond LCA and EOL
Beyond Life Cycle Assessment and End-of-Life modeling, effective feedback mechanisms are crucial for a truly circular design process. They provide real-time insights into a building's performance, such as energy efficiency and occupant comfort, allowing for continuous improvements. During construction, feedback on assembly ease and waste generation helps refine future designs and reduce waste. Additionally, tracking a building's adaptability and reusability ensures it can be easily repurposed or disassembled for component reuse. Economic feedback ensures that circular practices remain financially viable by analyzing costs and savings, while supply chain feedback optimizes material sourcing and reduces environmental impacts. Together, these feedback loops enable informed decision-making and support the creation of sustainable, resilient built environments.

Current LCA and EOL Tools - Essential but Limited:
While the tools described below are essential for establishing a baseline understanding of environmental performance, they are often insufficient to support the dynamic, multi-faceted feedback required for truly circular design. Their limitations, particularly in terms of data intensity, static nature, and integration with design processes, underscore the need for the more sophisticated, ML-enhanced feedback mechanisms discussed earlier. By augmenting these established tools with AI-driven approaches, we can create a more comprehensive and responsive feedback ecosystem.

Now, let's delve into the existing landscape of LCA and EOL tools:

SimaPro: A widely used, comprehensive LCA software that allows users to model and analyze the environmental impacts of products and services across their entire lifecycle. It includes extensive databases (e.g., ecoinvent) containing environmental data on a vast array of materials and processes. SimaPro supports various impact assessment methods (e.g., ReCiPe, TRACI) and facilitates sensitivity and uncertainty analysis.

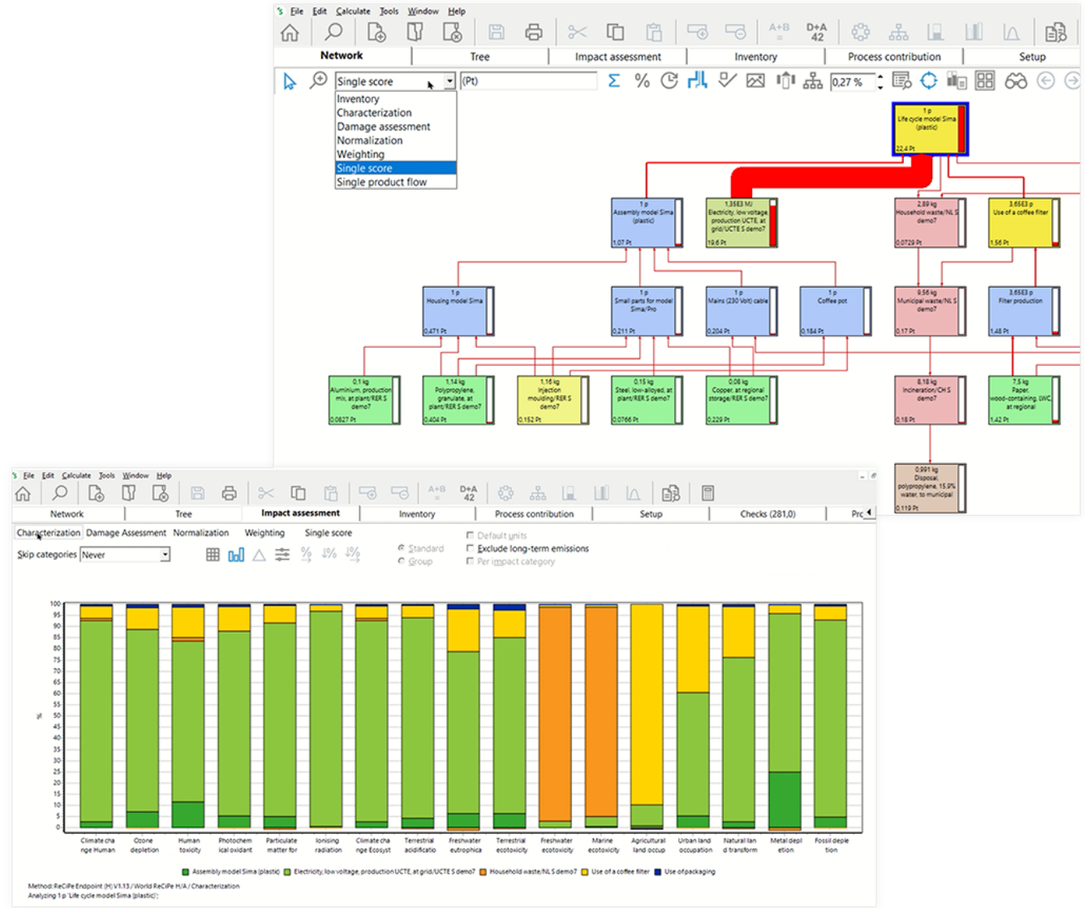


Figure 39: Sima Workplace
Source: https://simapro.com/

B. Literature Review
B.9.1 Existing Industry Tools

sphera: Another leading LCA software package offering similar functionalities to SimaPro. It provides tools for modeling complex product systems, conducting scenario analysis, and generating detailed environmental reports. GaBi also features a large database of environmental data and supports various impact assessment methodologies.

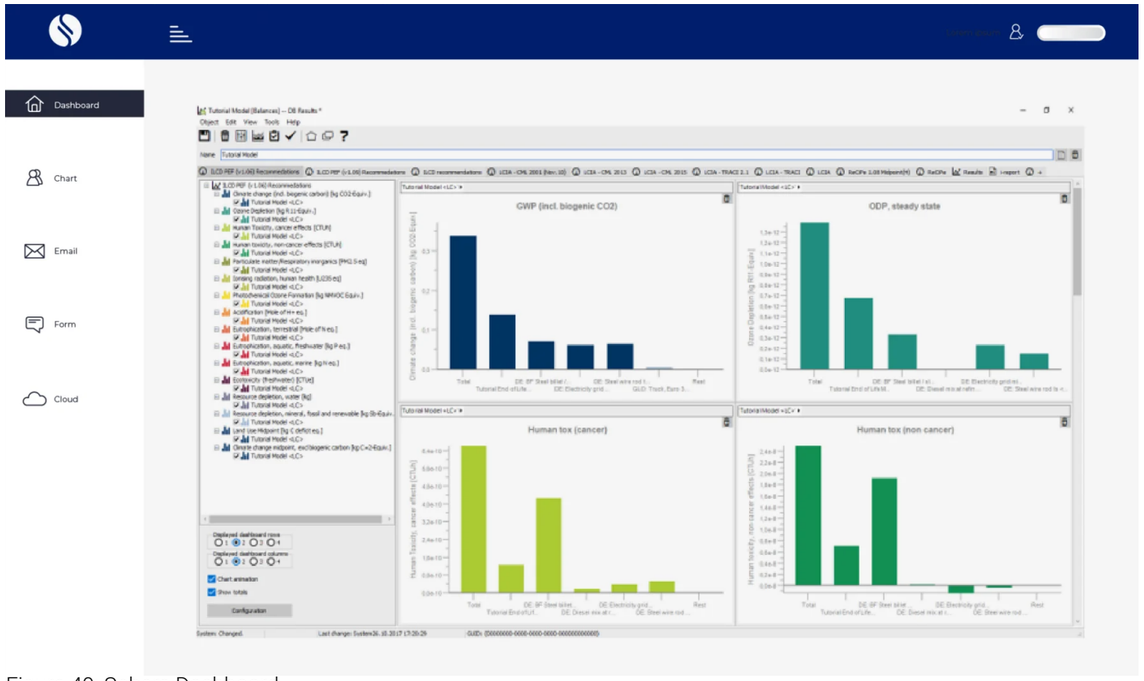


Figure 40: Sphera Dashboard
Source: https://sphera.com/

OpenLCA: An open-source alternative to SimaPro and GaBi, providing a flexible platform for LCA modeling and analysis. While it may have a steeper learning curve than commercial software, OpenLCA offers greater customization and transparency. It supports various databases and impact assessment methods and is particularly popular in academic and research settings.

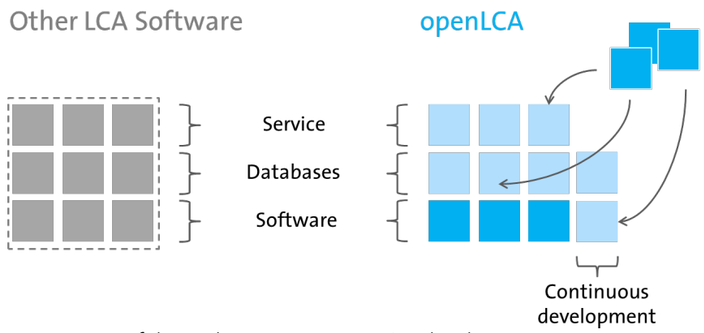


Figure 41:The open source nature of the tool promotes expansive development
Source: https://openlca.org

OneClickLCA: Specializing in the construction industry, OneClickLCA automates many of the complexities of building LCAs. It integrates with BIM software, drawing from established databases to produce quick assessments of embodied carbon, whole-life carbon, and other environmental metrics. As an automated tool, it may lack the customization of the options above.

Figure 42: One Click LCA
Source: https://carbonleadershipforum.org/

B. Literature Review
B.9.2 Design Integrated Circularity Assessment Tools

The core idea here is that we need to move away from assessing circularity as an afterthought and instead bake it directly into the design process.

RhinoCircular: A plugin deeply integrated within Rhinoceros 3D, a leading design software, offers streamlined circularity feedback directly within the design environment. This integration eliminates the need for platform switching, enhancing workflow efficiency. The plugin provides real-time feedback on design changes, enabling iterative adjustments and optimization, allowing designers to dynamically explore options and understand the circularity impact of their choices. RhinoCircular also actively supports material selection by assisting designers in choosing materials with strong circularity credentials, considering factors like recycled content, reusability, local sourcing, and end-of-life options. Furthermore, it visualizes material flows throughout the product lifecycle, helping designers identify potential waste streams, opportunities for material reuse, and areas for improving design for disassembly.

BHoM LCA Toolkit : The BHoM LCA Toolkit, built on the open-source Buildings and Habitats object Model (BHoM) framework, offers a flexible and adaptable solution for Life Cycle Assessment (LCA). Its open-source nature allows for customization to suit specific project needs and integration with various data sources. Seamless integration with Grasshopper, a popular visual programming language in the AEC industry, enables parametric modeling and analysis, allowing designers to explore a broader range of design options and their associated environmental impacts. The toolkit facilitates comprehensive LCA, evaluating environmental impacts across the entire life cycle of a product or building, from raw material extraction to end-of-life disposal. By providing detailed LCA data, the BHoM LCA Toolkit empowers designers to make data-driven decisions, prioritize sustainable choices, and minimize the overall environmental footprint of a project.

Design with Circulytics: Developed by the leading circular economy organization, the Ellen MacArthur Foundation, offers a holistic approach to assessing circularity performance at the company level. It encourages businesses to embed circular economy principles throughout their operations, rather than focusing solely on individual products or buildings. Circulytics provides a framework for measuring and tracking circularity performance across various aspects of a business, including material sourcing, product design, waste management, and business models, allowing companies to monitor their progress over time. Beyond assessment, the tool offers tailored recommendations for improvement, helping companies identify specific areas for enhancement and set concrete goals. Backed by the Ellen MacArthur Foundation, Circulytics is aligned with best practices in circularity, adding credibility and ensuring its effectiveness.

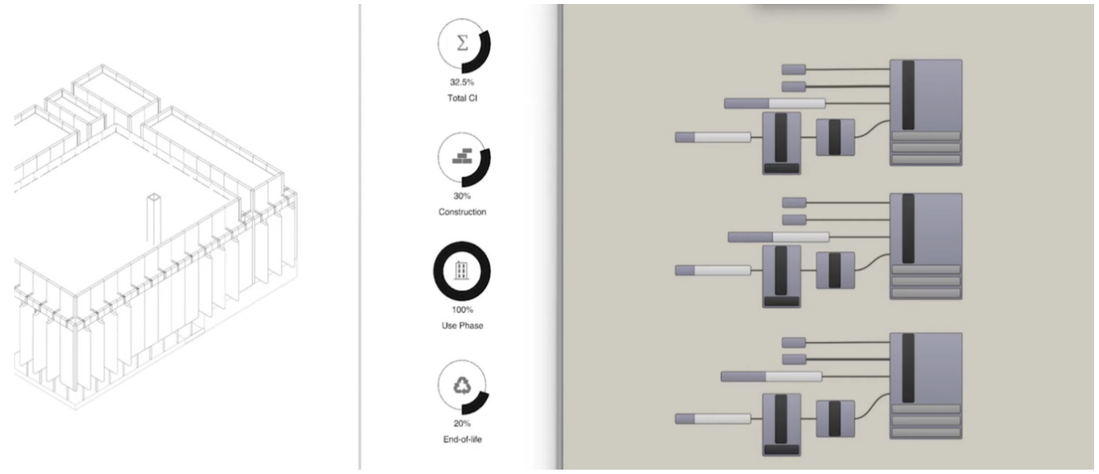


Figure 43:RhinoCircular user interface in Rhinoceros3D and Grasshopper. Circular Construction Lab, Cornell University. Source:Felix Heisel , Cameron Nelson 2024

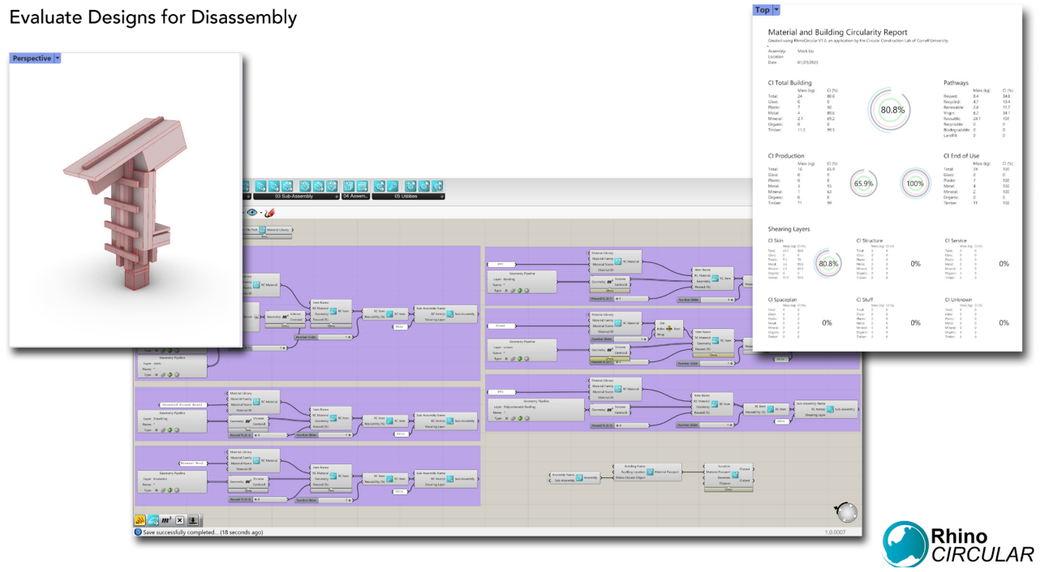


Figure 44: Design evaluation for disassembly
Source:https://labs.aap.cornell.edu/ccl/rhinocircular

B. Literature Review
B.9.2 Design Integrated Circularity Assessment Tools

Conclusion

The preceding sections emphasized the need for integrated, data-driven frameworks that embed circularity into the design and construction lifecycle, particularly for engineered wood products (EWPs). Machine learning, encompassing neural networks and reinforcement learning, can transform complex data into actionable insights, enhancing feedback mechanisms for better decision-making. Successful implementation requires robust Life Cycle Assessment (LCA) and End-of-Life (EOL) modeling, alongside a comprehensive understanding of feedback throughout a project's lifecycle. While LCA and EOL are crucial for measuring environmental impacts, they are only part of the necessary feedback system for a truly circular built environment. Continuous feedback loops beyond environmental metrics, including performance, constructability, adaptability, economic viability, and supply chain efficiency are essential for informed decisions.

Effective feedback mechanisms beyond LCA and EOL are crucial. These mechanisms provide real-time insights into building performance (e.g., energy efficiency, occupant comfort), allowing for continuous improvements. Construction feedback on assembly ease and waste generation refines future designs. Tracking adaptability and reusability ensures easy repurposing or disassembly for component reuse. Economic feedback validates the financial viability of circular practices, while supply chain feedback optimizes material sourcing and reduces environmental impacts.

Current LCA and EOL tools like SimaPro, sphera, OpenLCA, and OneClickLCA are essential for establishing a baseline understanding of environmental performance but are often insufficient to support the dynamic, multi-faceted feedback required for truly circular design. Their limitations in data intensity, static nature, and design process integration highlight the need for more sophisticated, ML-enhanced feedback mechanisms. Augmenting these tools with AI-driven approaches can create a more comprehensive and responsive feedback ecosystem.

B. Literature Review
B.10 Graph Neural Networks

We need a computational method that can think as fast as architects sketch and think in the same language the building is "drawn in". That method could be a neural network: A building module forms a node, and multiple nodes connect in various ways with edge connections.

In this way we retain two important pieces of information:

1.Node attributes. In our prototype a node stands for a building module which could be a whole room or cluster of rooms, but the same abstraction could just as easily represent a beam, column, or façade panel. The graph therefore has to log—clearly and compactly—the key physical properties of each element, its 3-D position, and the identifiers of the neighbors to which it connects. Graph Neural Networks are built precisely for such networks, so they let us keep the geometry and the relationships intact instead of flattening everything into lists or images (Battaglia et al., 2018).

2.Neighbor relationships. Every edge records how two elements interface otherwise influence one another, for example when building modules interface each other in some manner. By storing these links, the model preserves adjacency and joint type allowing a change in one module to ripple instantly to the ones it interacts with. Message-passing GNNs exploit exactly these neighbor links, aggregating information from connected nodes so that spatial effects emerge naturally during training (Kipf & Welling, 2017, Veličković et al., 2018).

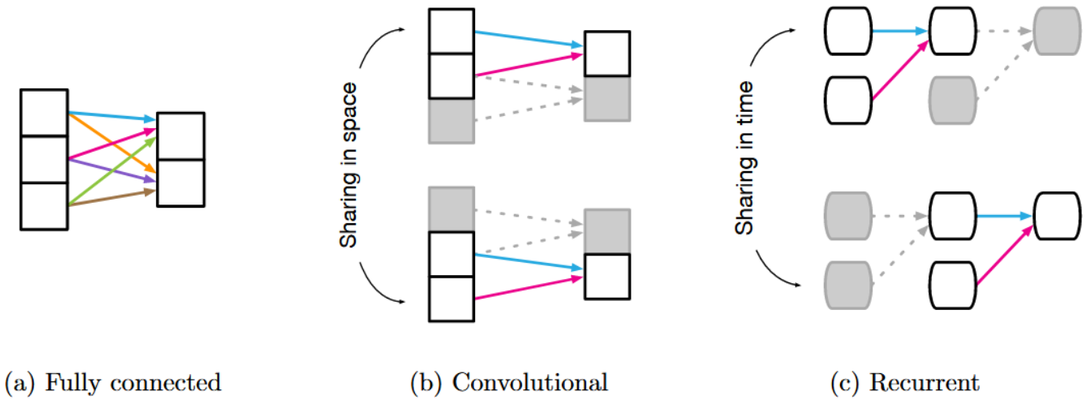


Figure 45: Reuse and sharing in common deep learning building blocks. (a) Fully connected layer, in which all weights are independent, and there is no sharing. (b) Convolutional layer, in which a local kernel function is reused multiple times across the input. Shared weights are indicated by arrows with the same color. (c) Recurrent layer, in which the same function is reused across different processing steps.
Source: Battaglia et al., 2018

B. Literature Review
B.10.1 GNNs As The Selected Computational Method

Four project-specific reasons for choosing Graph Neural Networks

1.One data container for geometry and life-cycle facts
A GNN treats every LVL room-module as a node and every joint as an edge, so the same graph tensor can carry spatial layout, Environmental Product Declaration figures, and manufacturing metadata side by side. That unified representation eliminates the data-fragmentation problem that plagues early-stage LCA and keeps the model's "view" of the project identical to the designer's drawing. Battaglia et al. show how this relational bias lets the network reason over structure–property links without flattening the geometry (Battaglia et al., 2018).

2.Sketch-speed feedback for iterative design
Message-passing updates scale roughly with the number of node connections, not with 3-D voxel counts or mesh density. After training, the prototype returns carbon and volume estimates in well under a second fast enough to plug straight into Grasshopper's live preview (Kipf & Welling, 2017).

3.Transparent, module-level explanations
Attention weights or integrated-gradient maps highlight the exact module or connector that blows the carbon budget, turning the network's verdict into an actionable design hint (Doshi-Velez & Kim, 2017; Veličković et al., 2018).

4.Data-efficient learning that future-proofs the tool
Weight-sharing across all nodes lets the network generalize from a modest LVL dataset to unseen layouts—and even to other bio-based materials once their EPDs are added—keeping sample requirements low (Hamilton, 2020; Wu et al., 2021).

B.10.2 Architecture

A Graph Neural Network, usually shortened to GNN, is a neural network that reads data stored as a graph. In a graph you have nodes, which can be anything from atoms to building elements, and edges, which say whether two nodes touch, support, or otherwise influence each other. A GNN learns by repeating three ideas that follow natural intuition: give every part a numeric "passport," let neighboring parts talk, then write a summary.

1.Encode the parts
Before learning can begin the raw description of each node and edge is turned into a vector of numbers. If the graph describes a building, a node vector may contain length, width, density, or material code and an edge vector may contain joint type or distance. This step is a small fully connected network that takes the raw values and returns a compact "passport" (Kipf & Welling, 2017).

B. Literature Review
B.10.2 Architecture

2.Pass messages between neighbors
The core of a GNN is a message-passing layer: each node collects messages from its immediate neighbors, blends them with its state, and stores the result. Stacking several layers lets information travel several hops so that a node eventually "knows" about a wider neighborhood.

A common formal description is:

$$\mathbf{h_i^{(k+1)}} = \gamma^{(k)}!\left(\mathbf{h_i^{(k)}}, \bigoplus_{j\in\mathcal{N}(i)} \phi^{(k)}\left(\mathbf{h_i^{(k)}}, \mathbf{h_j^{(k)}}, \mathbf{e_{ij}}\right)\right)$$

h^k_i is the vector held by node i after k rounds
N_i is the set of neighbors of i,
φ creates a message from a neighbor,
$\bigoplus$ is an aggregation such as a sum or a mean,
γ updates the node with the aggregated message (Gilmer et al., 2017).

You may meet different variations of this idea. A GCN averages normalised neighbour vectors (Kipf & Welling, 2017). A GraphSAGE samples a fixed number of neighbours to save memory (Hamilton, 2020). A GAT learns an attention weight that tells the network which neighbour matters most (Veličković et al., 2018).

3.Read out the answer
After a chosen number of message-passing rounds the network must turn many node vectors into something the task requires. For node classification it keeps a vector per node. For a whole-graph prediction it combines all node vectors with a permutation-invariant operation such as global mean, global sum, or an attention readout. The result goes through a small fully connected network that outputs the final numbers.

Why this matters

1.The passport step lets the network start with the raw facts you already have, such as area or density.

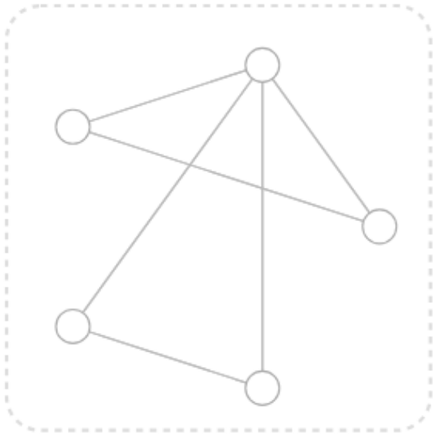
2.Message passing feels natural: each part looks at its neighbours, just as an engineer looks at connected elements.

3.The readout step guarantees that the final answer does not depend on how you happened to list the nodes.

LITERATURE REVIEW

B. Literature Review
B.10.2 Architecture

More Graph Neural Network Examples



V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

U Global (or master node) attributes
e.g., number of nodes, longest path

Figure 46: Graph elements
Source:https://distill.pub/2021/gnn-intro/

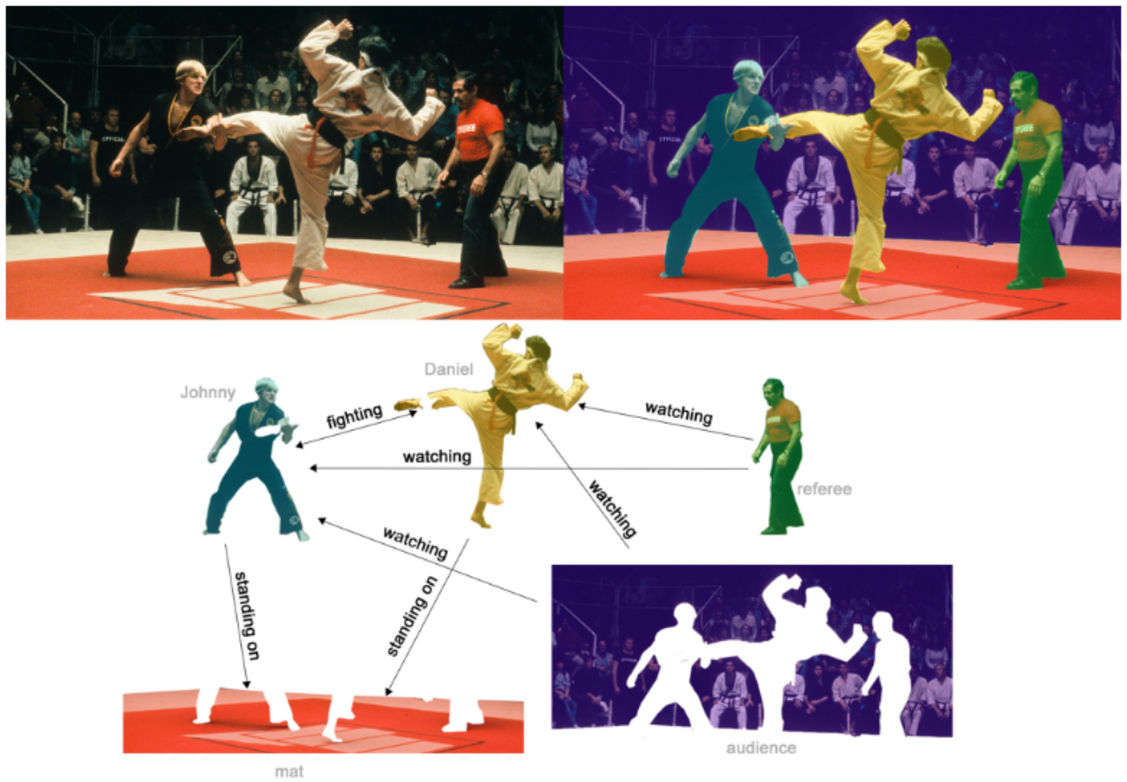
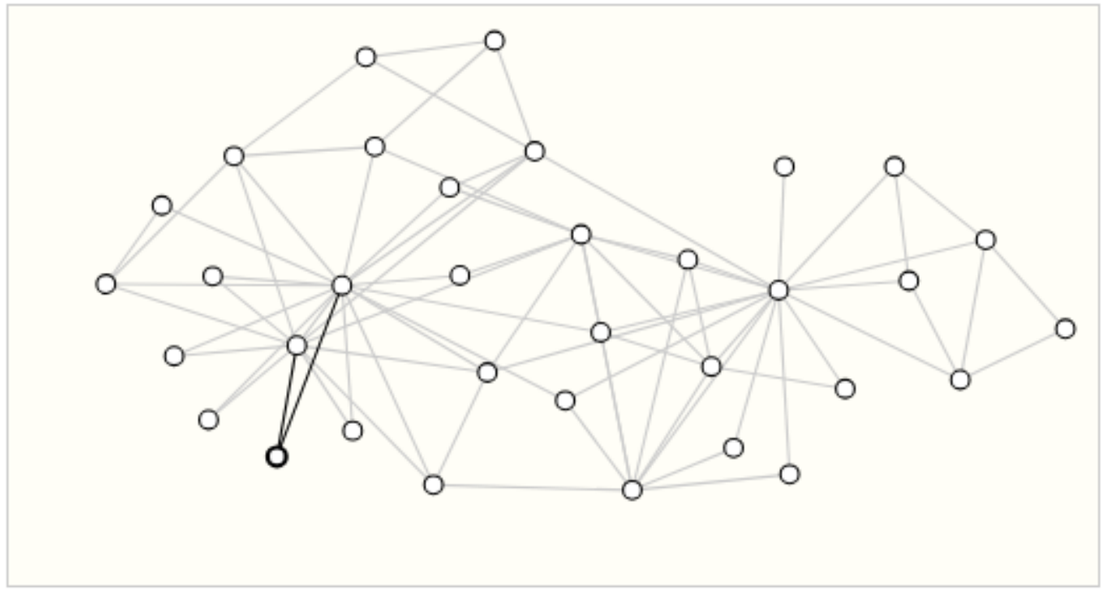


Figure 47: Edge level predictions, image becomes a graph by being seperated in layers and actions become edges.
Source:https://distill.pub/2021/gnn-intro/



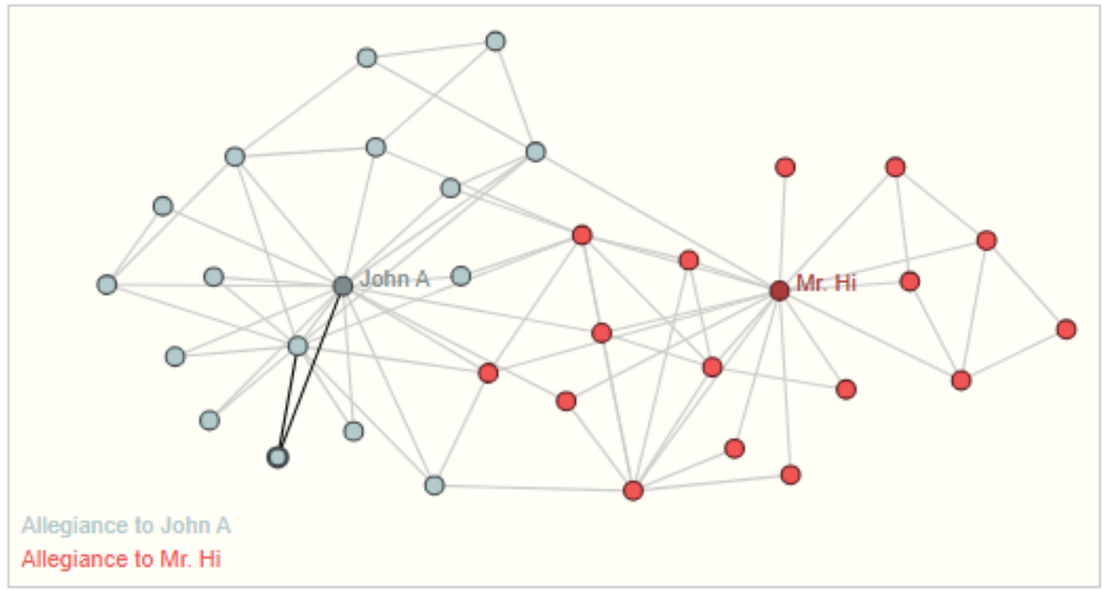


Figure 48: Node level predictions, the gnn predicts which of the nodes belong to each label.
Source:https://distill.pub/2021/gnn-intro/

B. Literature Review
B.11 Summary

The literature review reveals a central shortcoming in architectural design methodology: the absence of an integrated, explainable, and proactive life-cycle assessment (LCA) feedback loop at the conceptual stage. Crucial choices on form, structure, and materials are still taken with little real-time data on carbon, energy, or waste, reducing circularity to a late-stage compliance check rather than a formative driver (Kirchherr et al., 2017; Ghisellini et al., 2016).

This gap is especially problematic for bio-based materials such as Laminated Veneer Lumber (LVL), whose significant sustainability potential is scarcely captured by early-stage tools (Abed et al., 2022). As a result, architects lack the means to quantify environmental impacts precisely when it matters most—project inception.

Workflows remain fragmented: Environmental Product Declarations sit in external databases with little interoperability (Del Borghi, 2013); structural models are isolated from material inventories; and the few predictive models in use rarely connect to BIM environments (Adadi & Berrada, 2018; Doshi-Velez & Kim, 2017). Sustainability metrics thus enter only after designs are largely fixed, when alterations are costly or infeasible, leading to lost opportunities for low-impact innovation (Kirchherr et al., 2017; Ghisellini et al., 2016).

Addressing these issues demands an integrated, data-driven workflow embedding life-cycle and circularity metrics from the outset. The European Commission's building LCA model already links EPD-derived impacts with conceptual massing studies, proving early integration both possible and encouraged (European Commission JRC, 2018a). Yet the literature agrees that true circularity requires unifying geometric modelling, structural analysis, and environmental assessment into a single interoperable framework—so principles such as design for disassembly, reuse, and efficient material loops actively shape decisions rather than being audited post-facto (Pomponi & Moncaster, 2017).

Emerging work in machine learning and explainable AI points to a practical path for integration. Four GNN advantages match the review's needs:

GNNs capture building components or LVL modules and their relationships in a single graph, so geometric and LCA data coexist and the fragmentation of flat tables disappears (Battaglia et al., 2018; Del Borghi, 2013).

Trained GNN surrogates give sketch speed predictions because message passing scales with connections rather than detailed geometry, delivering the real-time feedback architects need (Kipf & Welling, 2017).

Feature attribution methods make GNN output transparent at component level, addressing concerns about black box models and showing which module features drive carbon results (Montavon et al., 2018; Molnar, 2022; Adadi & Berrada, 2018; Doshi-Velez & Kim, 2017).

By learning from relational data, GNNs prove naturally scalable, so a well designed model can handle larger assemblies or new material systems with little retuning (Veličković et al., 2018).

Together, these traits justify placing GNNs at the thesis's core, blending graph based representation with fast, intelligible LCA feedback rather than promising an all purpose fix.

LITERATURE REVIEW

B. Literature Review
B.11 Summary

In summary, the literature review justifies a set of focused research objectives and the overall scope of this thesis. It motivates the creation of a GNN-driven framework that directly integrates EPD-based environmental data with early-stage design models, provides explainable, real-time predictions of life-cycle performance, and remains adaptable to different design scenarios.**Rather than proposing a definitive solution, this work is positioned as an exploratory response to the challenges identified in prior studies, with objectives carefully chosen to advance the field's understanding of how such integration can be achieved.**

End Of Chapter B. Literature Review

Work Package 02 | Data Collection & Processing

C. Environmental Performance & Data Collection

Brief Summary

A validated EPD lookup table was produced (WP2) to consolidate all life-cycle intensity factors with a clear hierarchy of required, optional and default values. Missing data are managed through predefined fallbacks, ensuring complete coverage for transport, energy and material metrics.

Transport emissions combine real routing distances and vehicle logistics with haversine fallbacks to yield transparent formulas. Embodied-carbon and biogenic-sequestration intensities are then aggregated by summing over the life-cycle and end-of-life phases selected dynamically via user specific LCA and EOL scenarios.

Volume predictions from the graph-neural-network surrogate (WP3) are passed into a modular post-processing service (WP4) that multiplies each volume by the global intensities to produce real-time estimates of carbon, sequestration and renewable-fraction metrics.

Containerized services allow seamless addition of new manufacturers, waste streams, material families and geographic regions without altering core algorithms, while out-of-scope dimensions are explicitly bounded.

C. Enviromental Performance & Data Collection

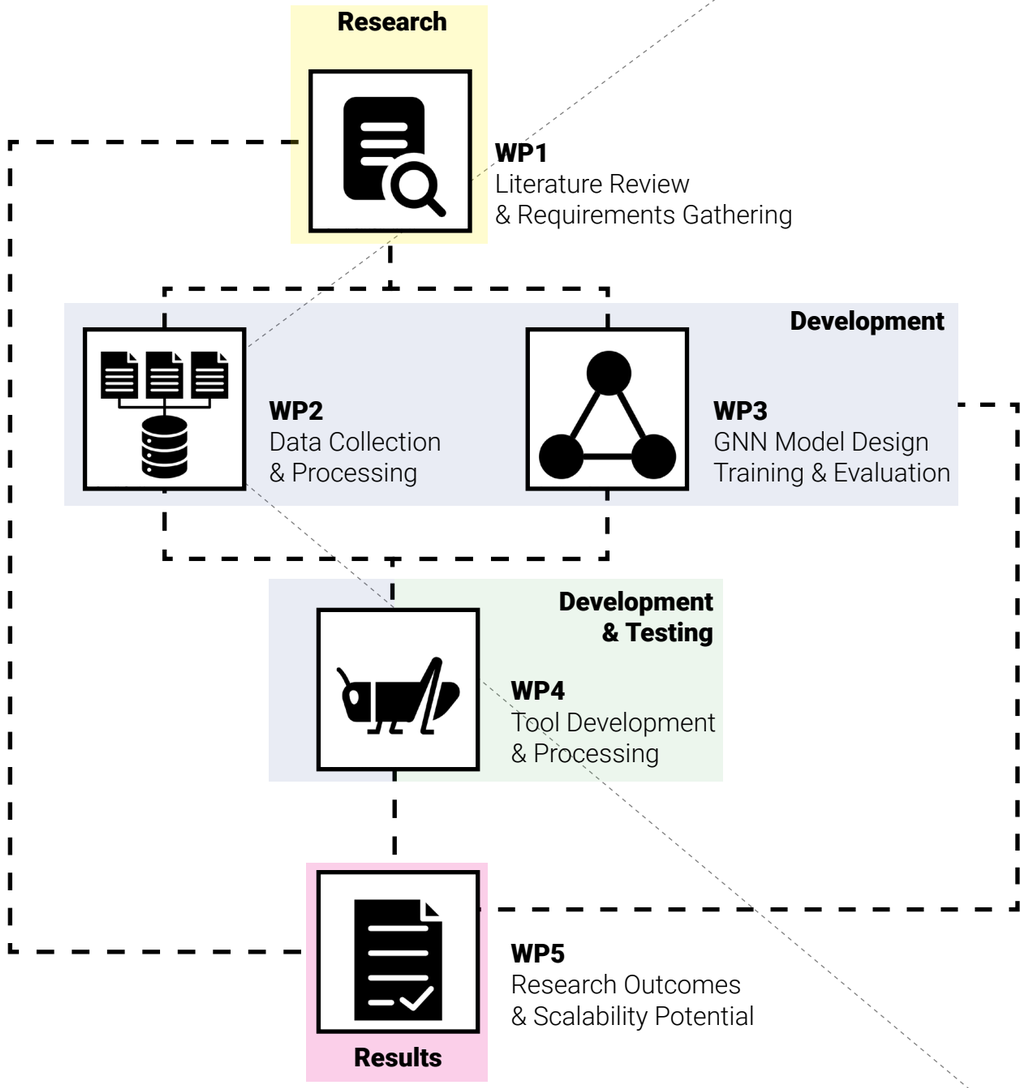


Figure 11: Work Packages Overview / Source: Author, 2025.

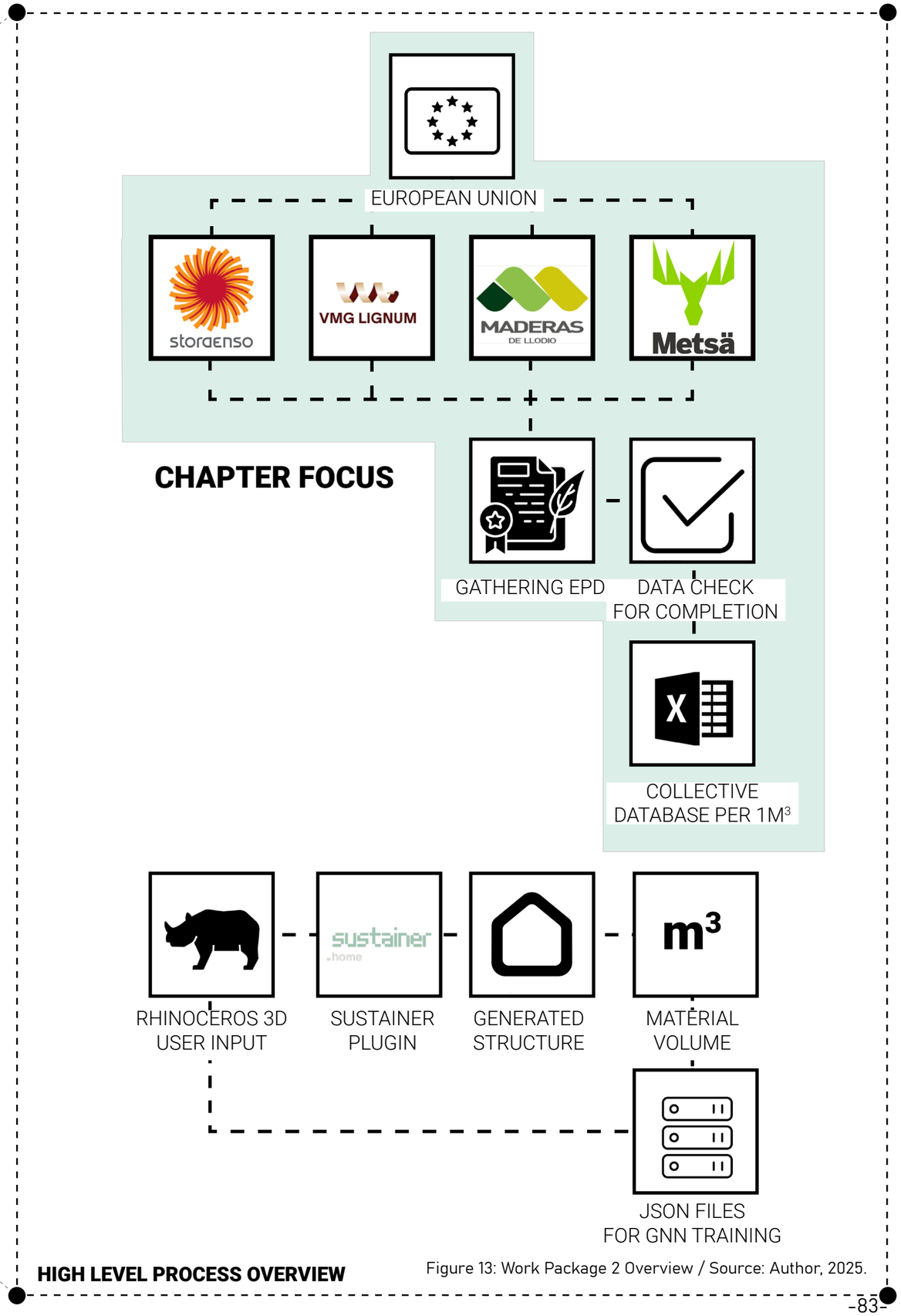


Figure 13: Work Package 2 Overview / Source: Author, 2025.

C. Environmental Performance & Data Collection

C.1 Selected Manufacturers

All four producers featured here supply laminated veneer lumber panels in the European Union under Environmental Product Declarations that conform to ISO 14025 and EN 15804. Each EPD declares a unit of one cubic metre of LVL, utilises predominantly softwood veneers bonded with phenol-formaldehyde adhesives, maintains third-party chain-of-custody certification (FSC® and/or PEFC) and carries CE marking in accordance with EN 14374. Transport, manufacturing and end-of-life modules (A1–A5, C1–C4) plus beyond-system credits (D) are addressed in every declaration, ensuring a cradle-to-grave perspective on environmental impacts.

Stora Enso
Production takes place at the Varkaus mill in Finland with an annual capacity of 85 000 m³ of LVL. Panels consist of 3 mm spruce veneers glued with phenolic and melamine-formaldehyde resin, achieving formaldehyde class E1. The division operates certified management systems for quality (ISO 9001), environment (ISO 14001), health and safety (ISO 45001) and energy (ISO 50001). All wood is sourced from Stora Enso's own certified forests .

Metsä Wood
Kerto® LVL is manufactured at two Finnish mills (Lohja and Punkaharju), combining 3 mm spruce veneers into a range of S-beams, Q-panels, L-panels and more. The company holds PEFC and FSC Chain-of-Custody certificates, plus ISO 9001, ISO 14001, ISO 45001 and ISO 50001 accreditation. Two end-of-life scenarios—energy recovery and recycling—are explicitly modelled in its EPD .

VMG Lignum Construction
Established in 2020, VMG Lignum's Lithuanian facility produces LVL-P (all veneers parallel) and LVL-C (up to 20 percent cross-glued) panels from locally sourced pine. Its EPD covers modules A1–A4, C1–C4 and D, with raw materials delivered entirely by road and end-of-life assumed as incineration with energy recovery. Chain-of-custody is ensured via FSC certification .

Maderas de Llodio (Grupo Garnica)
The Laudio, Spain plant manufactures radiata pine LVL across three product ranges (pine, deco and LVL) using sustainably managed wood within a 100 km radius. Its EPD applies economic allocation for co-products (chips, logs, bark), declares modules A1–A3 and C1–C4 plus D, and omits A4 transport. Packaging is accounted under A3 with wooden pallets and minimal plastic strapping.

To gather the information needed for the research and the data from the data requirements sheet in A.8, each EPD was reviewed to ensure it met the required criteria.

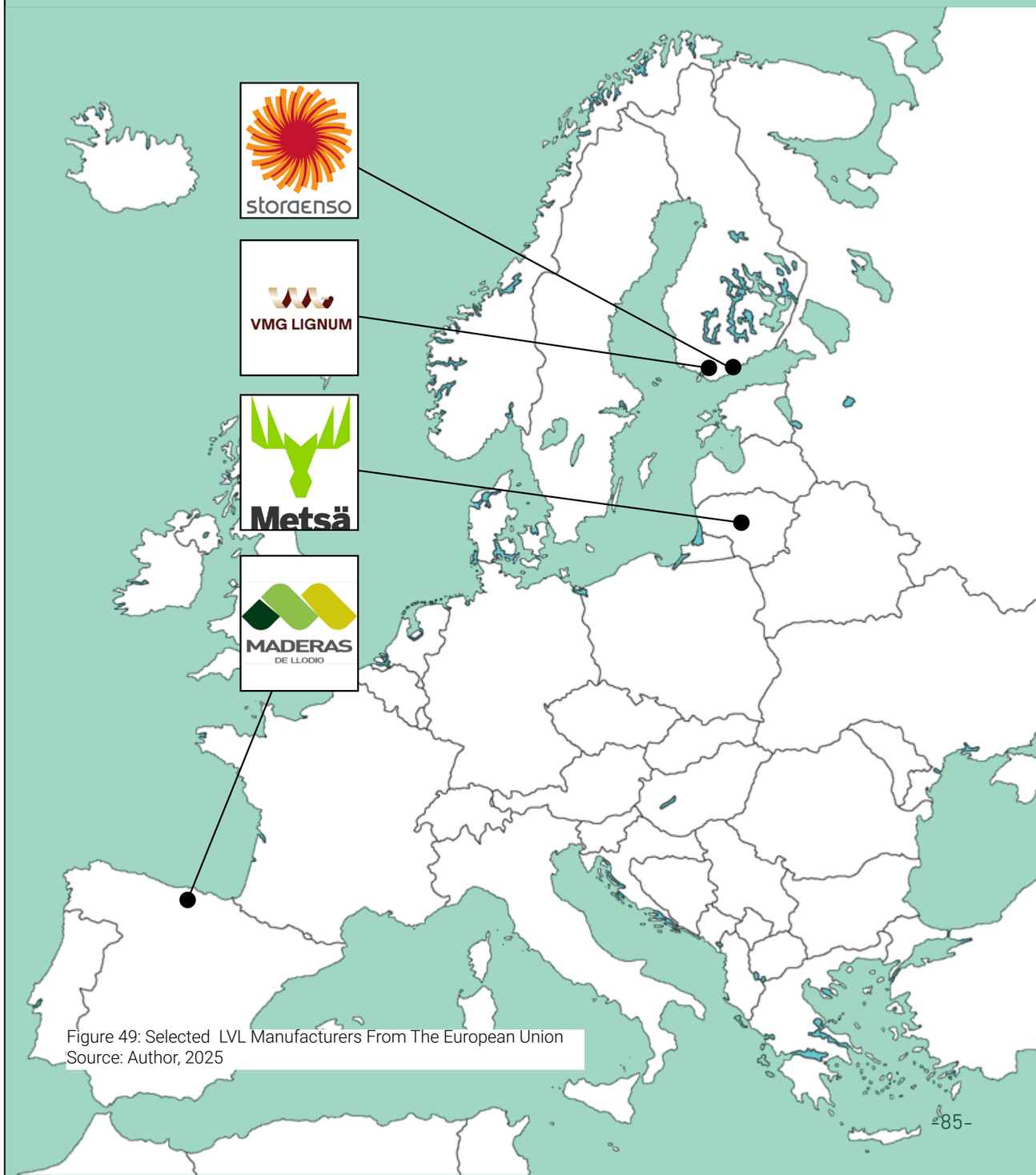


Figure 49: Selected LVL Manufacturers From The European Union
Source: Author, 2025

C. Environmental Performance & Data Collection
C.2 Comparative Analysis

When we view Stora Enso's life-cycle bars in figure 47 , it becomes immediately clear that the carbon stored in the raw wood (A1) overwhelms the small emissions from trucking and panel manufacture (A2 + A3). In effect, every kilogram of fossil-fuel CO_2 released during processing buys us many times over in tree-sequestered carbon. That strong "buffer" of biogenic storage sets the stage for the product's climate story.

Carrying that same lens into the full cradle-to-grave sequence in figure 48 reveals the critical role of how we handle material at end-of-life. Panels from Garnica and Metsä remain net carbon sinks, because their modest incineration impacts at C3 are easily offset by the retained forest carbon, and module D credits recover additional benefit. Stora Enso's larger combustion pulse starts to erode its initial advantage, yet it still finishes slightly negative. VMG Lignum, despite having the deepest carbon vault in its veneer, tips into net emissions once its heavy-energy burning is tallied. This shows that a strong cradle-gate sink can be completely undone if we choose an energy-intensive disposal.

Figure 49 break that down into fossil and biogenic contributions. Here we see two competing forces: the innate carbon capture of the wood and the fossil-fuel appetite of both production and end-of-life combustion. When the fossil slice grows—whether through high-temperature process heat or conventional incineration—it can overwhelm even a large wood-carbon reservoir.

It's important to emphasize that the new EPD data must be interpreted with caution. We proceed on the assumption that each manufacturer has reported their processes accurately and conscientiously, even though they themselves warn that these metrics are still preliminary due to limited experience.

In our comparative analysis of the four manufacturers, methodological differences inevitably emerge, especially in how each one calculates and reports manufacturing energy (MJ) which helps explain both minor discrepancies and the larger outliers. Finally, some producers may have selectively included or omitted certain production stages without clearly disclosing those choices in their documentation.

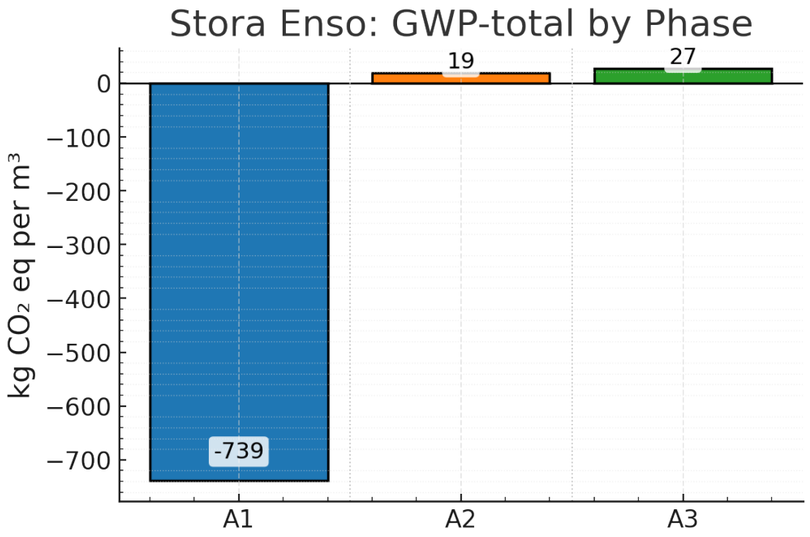


Figure 50: GWPtotal by LCA phase for Stora Enso
Source: Author, 2025

C. Environmental Performance & Data Collection
C.2 Comparative Analysis

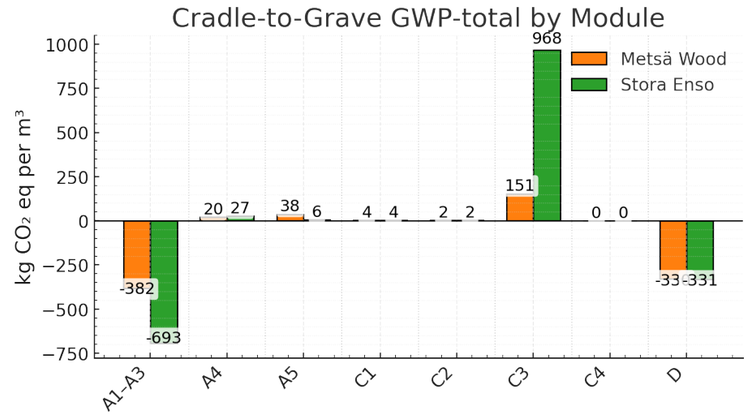


Figure 51: GWPtotal but for all of the lca phases
Source: Author, 2025

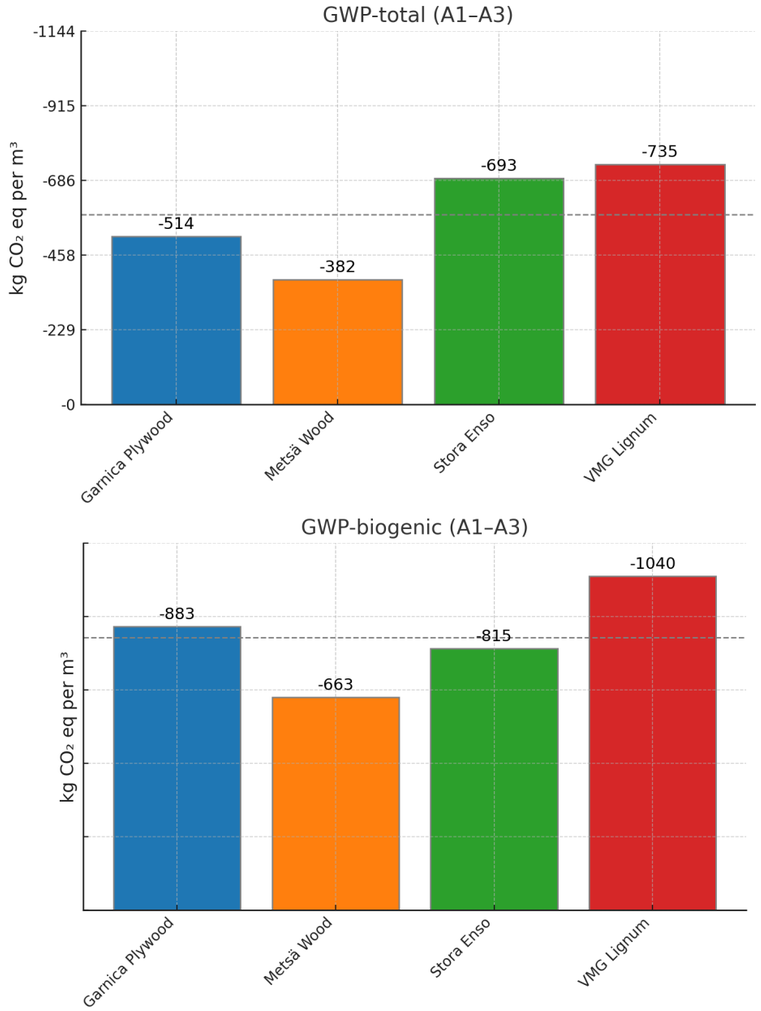


Figure 52: Performance of selected manufacturers.
Source: Author, 2025

C. Environmental Performance & Data Collection
C.3 Database Creation

Data Processing:

1.Curate a "clean" EPD lookup table (once, offline)
a)Store intensity factors only (e.g., kg CO_2-eq / m³, MJ / m³).
b)Tag each record provides warnings if information is missing by the selected manufacturer.

2.GNN inference
a)Feed the design to the trained model.
b)Output per node: predicted LVL volume (m³).
c)The GNN stops here—it never sees the EPD intensities.

3.Post-processing with the EPD table
a)Look up each node's manufacturer/product row and multiply:
b)Predicted volume × GWP intensity in kg CO_2-eq
c)Predicted volume × PERT / PENRT in MJ
d)Add transport (A4) by multiplying volume-derived mass with live OSM distance and a fixed kg CO_2-eq / t-km factor.
e)Overlay the chosen end-of-life scenario to modify C-modules and add/credit D.

By pushing EPD calculations after the machine-learning step, we keep the model small, fast, and geometry-focused while still grounding every output in third-party-verified environmental data.

C.4 In Case Of Missing Information

When building the consolidated EPD database for all four manufacturers, we follow a strict hierarchy of variables. Some are mandatory, when provided, we use the manufacturer's data directly. Others are optional, if a manufacturer omits an optional field, we insert a predefined default value so that all calculations can proceed smoothly. Finally, critical variables must always be present, if any of these are missing, the EPD record is deemed unacceptable and excluded from the database.

1. Non-Critical Variables

-Product Composition
-Packaging Information
-Product Weight & Mass Information

2. Optional Variables

-Chosen Method Of Transport (eg. truck type)
-Vehicle fuel consumption for truck transport
-Vehicle fuel consumption for ferry transport
-Emission factor for trucks
-Emissions factor for ferries
-Manufacturing Site Coordinated (if missing, must be filled by placeholder)

3. Critical Variables
-GWPtotal, GWPbiogenic, PERT,PENRT

C. Environmental Performance & Data Collection
C.4 Database Creation

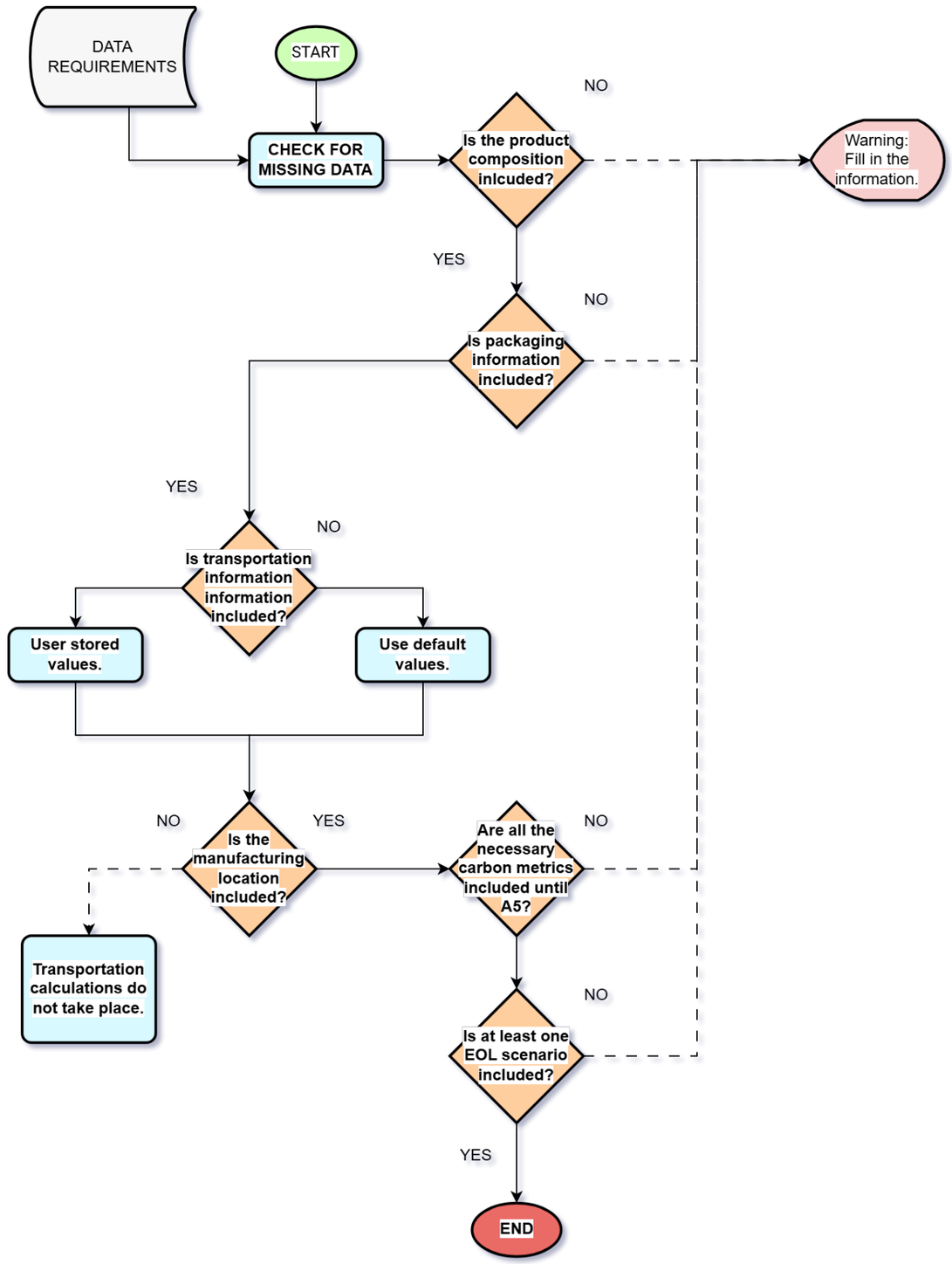


Figure 53: Data Aquisition Protocol
Source: Author, 2025

C. Environmental Performance & Data Collection
C.5 Calculating Transportation Emissions

Before beginning the detailed transport emission calculations, several key inputs must be in place. First and foremost, we require the total material volume, expressed in cubic metres (m³), as predicted by Work Package 3 (WP3). Equally important is the precise project location, typically provided as a full postal address, which allows for accurate geocoding and routing. In addition, the following data should be available:

1.Manufacturer and project site addresses (for geocoding to latitude/longitude).

2.Vehicle payload capacity (P), in m³ per trip.

3.Backhaul factor (β), representing the proportion of empty or partially loaded return journeys.

4.Vehicle-specific fuel consumption rates (f_T, in L/100 km) and emission factors (e_T, in kg CO_2/L fuel) for both truck and ferry segments.

5.Bulk density (ρ, in kg/m³) and transport-related emission coefficients (cells A17–E17) from the life-cycle inventory spreadsheet.

6.Access to the OpenRouteService (ORS) API, including a valid API key.

With these inputs secured, we can proceed through a structured, seven-step methodology to derive transport-related greenhouse-gas emissions.

Rather than relying on straight-line (“as-the-crow-flies”) approximations, we employ the OpenRouteService (ORS) API to obtain realistic road-network distances. Two key distances are retrieved:

1.Truck-only route
By invoking ORS’s “driving-hgv” profile with an explicit instruction to avoid ferries, we derive the heavy-goods-vehicle route distance, denoted . This value reflects legal restrictions on vehicle type and is rounded to three decimal places.

2.Complete route (truck + ferry)
A second ORS call without ferry avoidance yields , the full distance including any waterways.

We then compute the ferry segment as the difference between these two real-world distances:

$$d_{ferry} = d_{total} - d_{truck}$$

Because every route truly may includ=e both road and ferry legs, we always carry forward this difference—never artificially forcing to zero.

If the ORS service is ever unavailable (for instance, due to a missing API key or network interruption), a fallback calculation uses the haversine formula, , magnified by a detour factor of 1.25:

C. Environmental Performance & Data Collection
C.5 Calculating Transportation Emissions

$$d_{\text{fallback}} = 1.25 \times 2R \arcsin\left(\sqrt{\sin\left(\frac{\Delta\varphi}{2}\right)^2 + \cos\varphi_1 \cos\varphi_2 \sin\left(\frac{\Delta\lambda}{2}\right)^2}\right)$$

where R=6371 (the Earth’s mean radius) km, Φ denotes latitudes in radians, and λ longitudes in radians.

Vehicle Capacity and Trip Counts
Realistic logistics modeling requires more than distance; it also depends on how many loads must move and in what vehicle. Let V_{total} be the total material volume (m³) to be transported, and the usable volume per truck (m³). The number of round-trips, , is given by:

$$N = [V_{total}/P]$$

Every outbound loaded trip is typically followed by a partially-laden or empty return. We capture this with the backhaul factor β, so that the effective distance per loaded journey becomes

$$d_{eff} = d_{truck} \times (1 + \beta)$$

Values of β (e.g. 0.2 for a 20 % empty backhaul) should reflect observed fleet utilization. Additionally, different vehicle classes, rigid HGVs, articulated trucks, or vehicles meeting varying Euro-emission standards, exhibit distinct fuel consumption, f_T(L / 100 km), and emission factors e_T, (kg CO_2 / L fuel). These parameters feed directly into our emissions calculation.

Segment-Specific Emission Factors
Transport emissions arise from the interaction of distance, material characteristics, and vehicle efficiency. From our EPD database, we extract:

-Bulk density p (kg / m³) (cell A17).

-Truck coefficients: fuel factor c_{truck} in B17 and e_{truck} emission factor (D17).

-Ferry coefficients: fuel factor c_{ferry} in C17 and e_{ferry} emission factor (E17).

-Special case for “Metsa Wood”: combined GWP-fossil factor (F14), used as a single, per-m³ transport emission intensity.

For the truck segment of length d_{truck} , the per-cubic-metre emission factor is:

$$\mathrm{EF}_{\text{truck}} = d_{\text{truck}} \times \frac{\rho}{1000} \times c_{\text{truck}} \times e_{\text{truck}}$$

C. Environmental Performance & Data Collection
C.5 Calculating Transportation Emissions

Similarly, the ferry component is:

$$\mathrm{EF}_{\mathrm{ferry}} = d_{\mathrm{ferry}} \times \frac{\rho}{1000} \times c_{\mathrm{ferry}} \times e_{\mathrm{ferry}}$$

These sum to a combined factor,

$$\mathrm{EF}_{\mathrm{total}} = \mathrm{EF}_{\mathrm{truck}} + \mathrm{EF}_{\mathrm{ferry}}$$

unless a material-specific factor overrides (e.g. F14 for “MetsaWood”).

Aggregating to Total Emissions
We now scale emissions by both the number of trips and transported volume. Two equivalent formulations are possible:

1.Trip-based:
compute per-trip emissions for the truck leg,

$$\mathrm{EF}_{\mathrm{truck,trip}} = d_{\mathrm{truck}} \times \frac{f_T}{100} \times e_T \times \Big(1 + \beta\Big)$$

and then total truck emissions:

$$\mathrm{TE}_{\mathrm{truck}} = N \times \mathrm{EF}_{\mathrm{truck,trip}}$$

The ferry leg is analogous, and finally:

$$\mathrm{TE}_{\mathrm{total}} = \mathrm{TE}_{\mathrm{truck}} + \mathrm{TE}_{\mathrm{ferry}}$$

2.Volume-based: multiply total volume by the combined per-m³ factor,

$$\mathrm{TE}_{\mathrm{total}} = V_{\mathrm{total}} \times \mathrm{EF}_{\mathrm{total}}$$

Either approach yields the same result when capacity and backhaul are properly embedded.

Implementation Workflow
The code developed for the calculation of transportation emissions, executes the following steps in sequence:
1.Geocode manufacturer and project addresses to latitudes/longitudes.
2.Request ORS distances d_{truck} and d_{total}, compute d_{ferry}.
3.Specify payload P, backhaul β and vehicle specs f_T, e_T.
4.Compute trip count N.
5.Extract ρ and transport coefficients from the spreadsheet.
6. Calculate per-m³ factors EF_{truck} and EF_{ferry}.
7. Aggregate across trips or by volume to yield total transport emissions TE_{total}.

By anchoring distances in real routing data, accounting for vehicle logistics, and applying transparent MathML-encoded formulas, this methodology provides a rigorous estimate of transport-related CO_2-equivalent emissions in the life-cycle assessment.

C. Environmental Performance & Data Collection
C.6 Calculating Embodied & Sequestrated Carbon

In this chapter we describe how to derive the total embodied carbon (GWP) and biogenic carbon sequestration (CS), as well as the renewable ratio, from life-cycle inventory data. The procedure comprises four main parts: identifying prerequisites, summing LCA indicators, allocating to modules, and outlining the implementation workflow.

Before beginning, the following information must be at hand:

1.A prediction of total material volume (Vtotal in m³) as provided by WP3.

2.The EPD database from WP2, containing the LCA data, including the sheet name (manufacturer name) to use.

3.An LCA scenario index (0, 1, or 2) that determines which columns get summed and more precisely:
a) if LCA_ID=0, then we apply the cradle to gate scenario for calculations
b)if LCA_ID=1, then we apply the cradle to grave scenario and
c) if LCA_ID=2, then we apply the cradle to grave and beyond scenario

4.Similarly with LCA we also have an end-of-life filter value (eol = 0–3) selecting the correct back-end rows.
a) if EOL_ID=0, then we apply the incineration scenario
b)if EOL_ID=1, then we apply the reuse scenario
c) if EOL_ID=2, then we apply the recycling scenario
d) if EOL_ID=3, then we apply the landfill scenario

Basically the selection of IDs for LCA and EOL sets which rows of values we need from the data, from the corresponding LCA Phases (A1-D) for the GWPtotal,GWPbiogenic,PERT and PENRT values which are documented for 1m³ of Laminated Veneer Lumber.

5. From the trained GNN model from WP3, a grasshopper component is created to load the trained model (WP4) which provides material volumes predictions in real time the per building module in our design we get a list of individual module volumes (v1, v2, …, v3 in m³) for allocation and to use in coordination with our database.

Active Phase Set and Summation
Define two phase-sets:

1.L = the life-cycle stages selected by LCA_ID

2.E = the end-of-life stages selected by EOL_ID

Then the full summation set is:

$S = L \cup E$

C. Environmental Performance & Data Collection
C.6 Calculating Embodied & Sequestrated Carbon

For each phase s ∈ S, the database gives four per-unit intensities (all in kg CO_2 eq/m³):

e_s = embodied-carbon intensity
b_s = biogenic-sequestration intensity
p_s = renewable-input intensity
q_s = non-renewable-input intensity

We then form the global intensity totals:

Total embodied carbon

$$\mathrm{GWP\_total} = \sum_{s \in S} e_s$$

Total sequestrated carbon

$$\mathrm{CS\_total} = \sum_{s \in S} b_s$$

Total renewable inputs

$$\mathrm{PERT\_total} = \sum_{s \in S} p_s$$

Total non-renewable inputs

$$\mathrm{PENRT\_total} = \sum_{s \in S} q_s$$

Finally, **the renewable-fraction ratio is**

$$\mathrm{ratio} = \frac{\mathrm{PERT\_total}}{\mathrm{PERT\_total} + \mathrm{PENRT\_total}}$$

Allocation to Volumes
With global totals in hand, each predicted volume v_i (m³) receives:

Embodied Carbon entry for i:

$$\mathrm{EC}_i = \mathrm{GWP\_total} \times v_i$$

Sequestrated Carbon entry for i:

$$\mathrm{CS}_i = \mathrm{CS\_total} \times v_i$$

Renewable credit entry for i:

$$\mathrm{RR}_i = \mathrm{ratio} \times v_i$$

C. Environmental Performance & Data Collection
C.6 Calculating Embodied & Sequestrated Carbon

Implementation Workflow

1.From LCA_ID, select the life-cycle stage set L.
2.From EOL_ID, select the end-of-life set E.
3.Form S=L ∪ E.
4.Retrieve intensities e_s,p_s,b_s,q_s for each s ∈S.
5.Sum as above to get GWP_{total},CS_{total},$PERT_{total}$,$PENRT_{total}$.
6.Compute ratio.
7.For each volume vi, calculate EC i,CS i , and RRi via multiplication.

This structure makes it clear that which phases are summed depends dynamically on your two scenario indices, and that the same simple summation formulas apply regardless of that choice.

Important notice: When the selected LCA scope terminates before the onset of any C-phase processes (for example, cradle-to-gate A1–A3 or cradle-to-grave A1–B7), no C1–C4 or downstream end-of-life stages are present in the inventory.

In such cases, any end-of-life (EOL) selection is automatically disregarded, since incineration, reuse, recycling or landfill impacts cannot be applied when C-phase data are absent. Only LCA scopes that extend into C-phase (C1–C4) activate the EOL_ID and include the corresponding end-of-life contributions.

Mandatory impact category indicators according to EN 15804+A2									INCINERATION				
Indicator	Unit	A1	A2	A3	A1-A3	A4	A5	B1-B7	C1	C2	C3	C4	D
GWP total	kg CO2 eq.	-6.93E+02	1.88E+01	2.70E+01	-6.93E+02	2.73E+01	6.08E+00	0.00E+00	4.01E+00	2.22E+00	9.68E+02	0.00E+00	-3.31E+02
GWP-biogenic	kg CO2 eq.	-8.15E+02	1.21E-02	5.76E-01	-8.15E+02	1.08E-02	7.50E-04	0.00E+00	6.98E-04	8.81E-04	8.17E+02	0.00E+00	-9.26E-01
Mandatory impact category indicators according to EN 15804+A2													
Indicator	Unit	A1	A2	A3	A1-A3	A4	A5	B1-B7	C1	C2	C3	C4	D
PERT	MJ	1.72E+04	7.39E+00	6.19E+02	1.72E+04	5.64E+00	3.15E-01	0.00E+00	3.07E-01	4.59E-01	-7.58E+03	0.00E+00	5.50E+02
PENRT	MJ	6.75E+03	3.54E+02	2.72E+03	6.75E+03	4.71E+02	3.83E+00	0.00E+00	5.79E+01	3.83E+01	-1.74E+03	0.00E+00	-6.76E+03

	REUSE				
	C1	C2	C3	C4	D
GWP total	4.01E+00	2.22E+00	8.17E+02	0.00E+00	-1.17E+02
GWP-biogenic	6.98E-04	8.81E-04	8.17E+02	0.00E+00	-1.20E-01

	RECYCLING				
	C1	C2	C3	C4	D
GWP total	4.01E+00	2.22E+00	8.23E+02	0.00E+00	-1.76E+02
GWP-biogenic	6.98E-04	8.81E-04	8.17E+02	0.00E+00	-1.77E-01

	LANDFILL				
	C1	C2	C3	C4	D
GWP total	4.01E+00	2.22E+00	0.00E+00	1.09E+03	-4.81E-02
GWP-biogenic	6.98E-04	8.81E-04	0.00E+00	1.08E+03	-1.59E-04

Figure 54: Storing Information and the seperate EOL scenarios.
Source: Author, 2025

C. Environmental Performance & Data Collection

C.7 Scalability Outside Of The Scope

This framework is built to grow in both data coverage and regional reach without altering its core methodology. By following the protocols established in earlier chapters, new manufacturers, additional environmental variables and entirely new material categories can be integrated smoothly. Key procedural steps are:

1.Ingest each new manufacturer's verified life-cycle data into the centralized lookup table, applying the same hierarchy of mandatory, optional and critical fields.

2.Add any extra variables,such as waste generation, packaging types or novel emissions factors,by introducing new columns in the inventory schema and running them through existing validation checks.

3.Incorporate new material families, whether other wood products or non-wood construction materials, by feeding their per-unit intensities into the same LCA and end-of-life selection logic.

4.Expand geographic coverage simply by supplying new origin and destination coordinates to the transport calculator; the routing and emission routines remain unchanged.

C.8 Conclusions

Requirements and functional specifications for live, explainable LCA feedback were defined in WP1 by aligning policy objectives, circular-economy targets and computational capabilities. A unified, validated EPD lookup table with mandatory, optional and critical intensity fields was produced in Work Package 2. Using these data, transport emissions were quantified via real routing distances and vehicle logistics through calculation formulas. Embodied-carbon and biogenic-sequestration intensities were consolidated by summing life-cycle phase contributions dynamically selected by the user-chosen LCA and End Of Life scenarios.

Volume predictions that are generated by the WP3 graph neural network, are the core numerical inputs WP2 needs to use the created database and start producing results. The modular, containerized architecture allows the possible addition of new manufacturers, waste streams, material families and geographic regions by extending the lookup table or updating routing parameters without altering core algorithms.

Out-of-scope elements like dynamic carbon-sequestration modelling, real-time supply-chain variability and social LCA—were explicitly delineated to ensure methodological clarity and reproducibility. Prototype performance, interface usability and cross-context generalizability will be evaluated in WP5. In sum, life-cycle assessment is reframed from a retrospective compliance exercise into a proactive, data-driven design tool.

End Of Chapter C. Environmental Perfomance & Data Collection

Work Package 02 | Data Collection & Processing

D. From Design To Graph

Brief Summary

This chapter outlines the process of converting modular architectural layouts into graph-structured datasets for training and evaluating a Graph Neural Network. We began by differentiating between richly detailed Training Designs complete with Sustainer-generated geometry and exact material volumes and lightweight User Designs defined only by 3D boxes and planar-curve openings. An automated Grasshopper–Python pipeline then isolates each design in Rhino, extracts node and edge features, and serializes them into JSON files that align with standard GNN data formats. Strategic sampling via Latin Hypercube ensured 600 diverse training scenarios, while an additional set of out-of-distribution assemblies tested the model's ability to generalize to larger, more complex configurations. Together, these steps produce a scalable, heterogeneous corpus that enables the GNN to learn accurate volume predictions from minimal design inputs and to perform robustly on novel layouts.

D. From Design To Graph

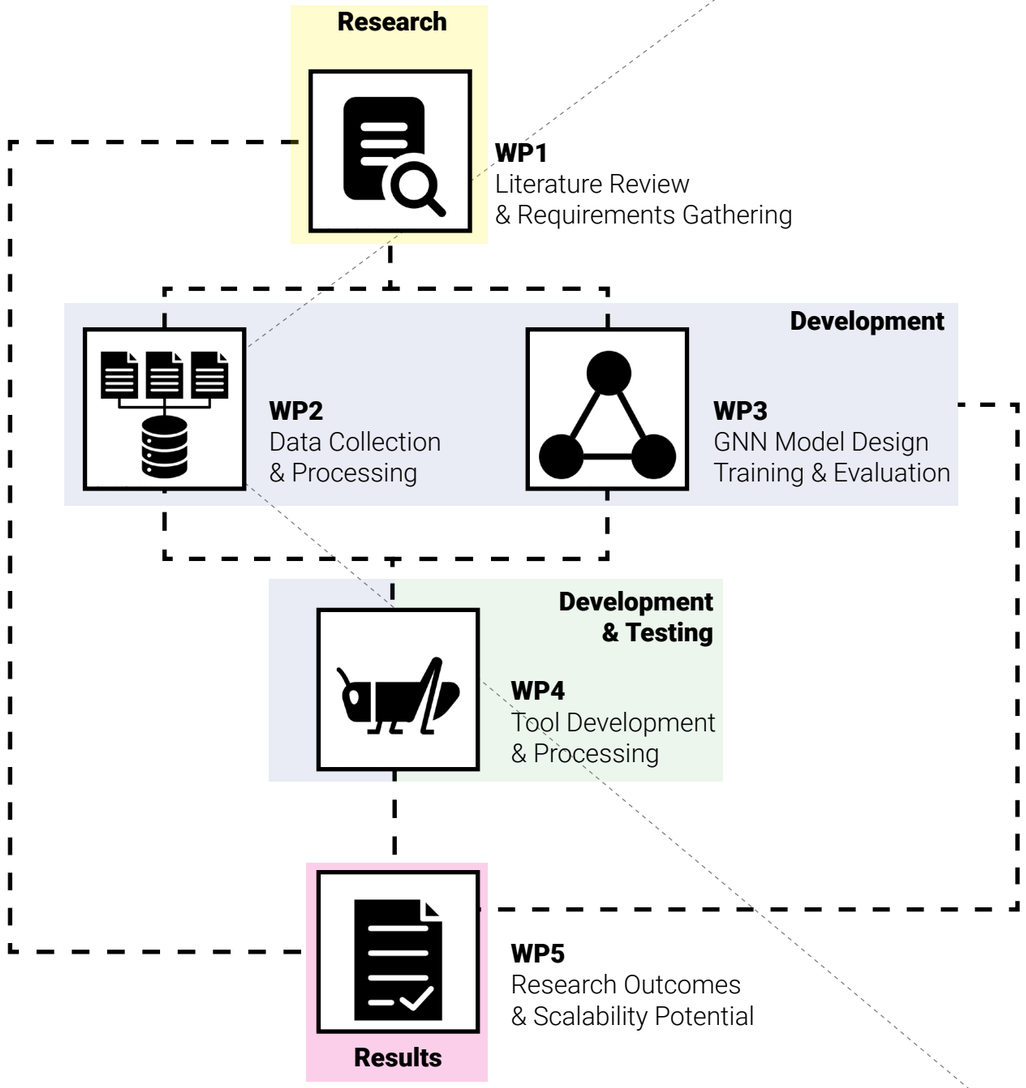


Figure 11: Work Packages Overview / Source: Author, 2025.

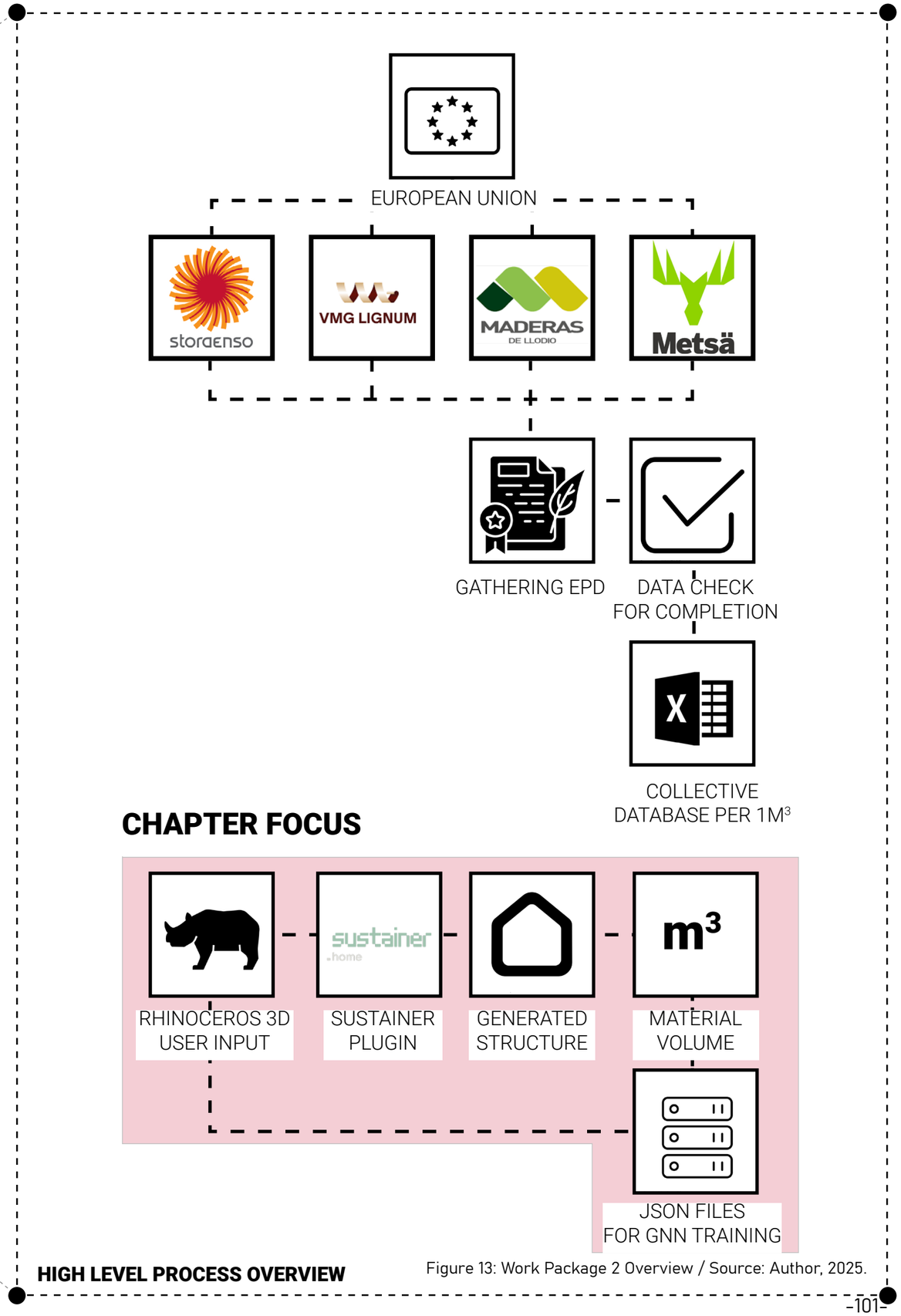


Figure 13: Work Package 2 Overview / Source: Author, 2025.

D. From Design To Graph

D.1 A Design In The Scope Of The Research

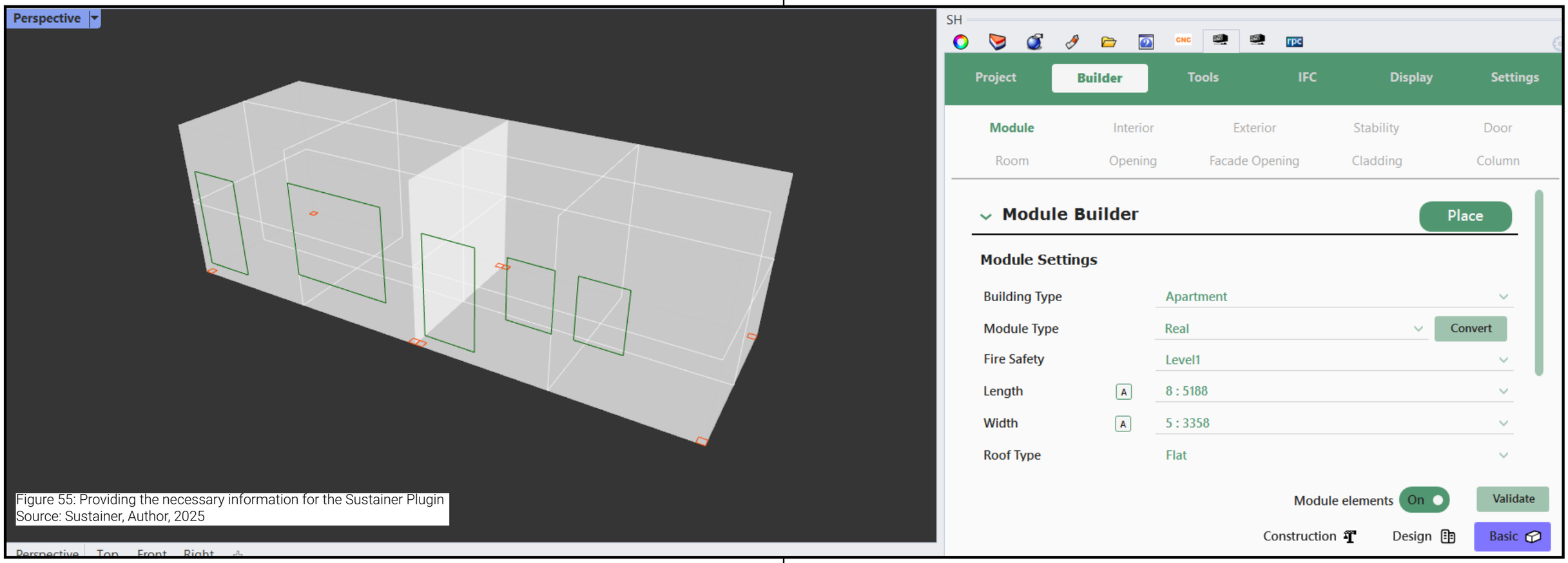


Figure 55: Providing the necessary information for the Sustainer Plugin
Source: Sustainer, Author, 2025

For this research, we distinguish between two definitions of “design.”

Training Design
This is the representation we use to generate our training dataset. It must encapsulate all the necessary geometric details and material-quantity parameters in a machine-readable format.

User Design
This is the input that end-users will create when they interact with our tool, seeking assistance to evaluate their own designs.

Focusing on the Training Design
Our immediate goal is to formalize the Training Design. It needs to capture:

1.Geometry: The overall shape and arrangement of each building module, modeled as 3D boxes.

2.Openings: Windows, doors, or other voids, represented as closed planar curves.

3.Material Quantification: The volume of material within each module, accounting for both solid regions and cut-outs.

Hypothesis
We propose that a Graph Neural Network (GNN) can learn to predict material volumes from these representations. Specifically, the network must:

1.Measure Module Sizes
Understand the dimensions of each 3D box.

2.Assess Openings
Quantify the area and, by extension, the volume removed by each planar curve.

3.Interpret Spatial Relationships
Grasp how modules are positioned relative to one another and how their interfaces influence overall material usage.

By integrating these factors, the GNN should accurately infer the total material volume across an assembly of modules.

D. From Design To Graph

D.1 A Design In The Scope Of The Research

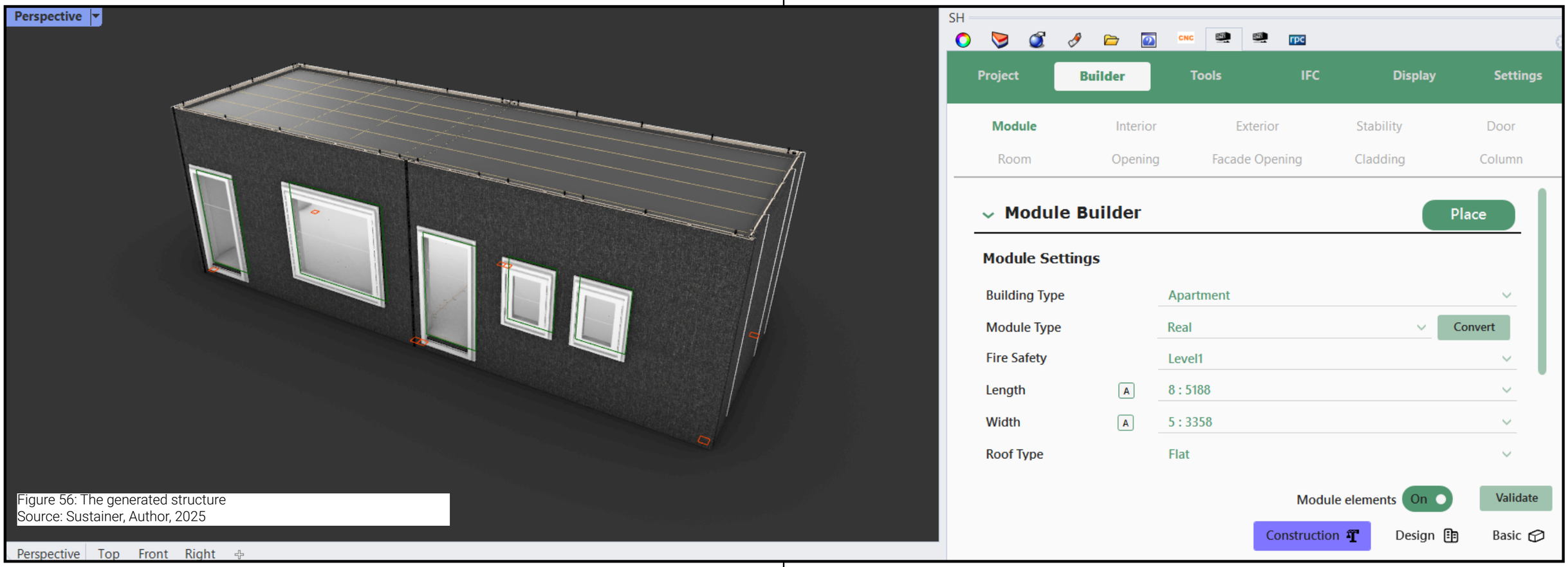


Figure 56: The generated structure
Source: Sustainer, Author, 2025

Industry Collaboration
To help us generate high-fidelity training examples, we're partnering with Sustainer, whose toolkit streamlines the process:

1.Input: We provide tour 3D boxes and define their openings through Sustainer's user interface.

2.Output: Their system automatically generates a fully realized, manufacturable geometry, complete with structural analysis and fire-safety compliance—ready for 1:1 fabrication.

This workflow not only supplies ground-truth volumes but also ensures that our training data reflect real-world engineering constraints.

Module Size Directory
To ensure consistency in our training data, each 3D box is drawn from a pre-defined size directory. Rather than allowing arbitrary dimensions, we limit module lengths, widths, and heights to a curated set of options—e.g., lengths from 1.0 m up to 10.0 m (in 0.5 m increments), corresponding material volumes already computed.

While the Sustainer plugin itself supports fully custom dimensions, for the purposes of this study we strictly adhere to this fixed catalogue. This constraint:

Standardizes Inputs

Every training example shares a common basis of discrete size choices, simplifying the GNN's learning task.

Bounds the Design Space
Prevents combinatorial explosion of possible geometries, focusing on a representative yet manageable range of module configurations.

Facilitates Ground-Truth Volume Lookup
Pre-computed material quantities for each standard size can be retrieved directly, streamlining data generation.

By combining these fixed-size modules—each with its associated opening profiles—we can systematically assemble diverse building configurations while maintaining control over the underlying geometry and volume parameters.

D. From Design To Graph
D.1 A Design In The Scope Of The Research

Material Layer Separation & LVL Object Isolation
Once a building configuration is generated via the Sustainer plugin, every geometry is automatically sorted into material-dependent layers, and each primitive (walls, floors, openings, etc.) is assigned a unique object ID. This structured layering allows us to programmatically query any subset of the model by material type.

For our research, we concentrate exclusively on the Laminated Veneer Lumber (LVL) components. To do this, we:

1.Filter by Layer
Extract only those objects residing on the LVL layer.

2.Collect Geometry Metadata
For each LVL object ID, retrieve dimensions, centroids, and topological relationships.

3.Compute Volumes
Sum the solid volumes of all LVL objects to obtain both:

a)Total LVL Volume for the entire building assembly

b)Per-Module LVL Volume by grouping objects according to their originating 3D-box module ID.

By isolating LVL elements in this way, we can precisely track how different module sizes, opening configurations, and module-to-module interfaces affect the required LVL material. In subsequent experiments, these per-module LVL volumes will reveal how structural connections—and variations in module arrangement—drive changes in material usage under our GNN's predictive framework.

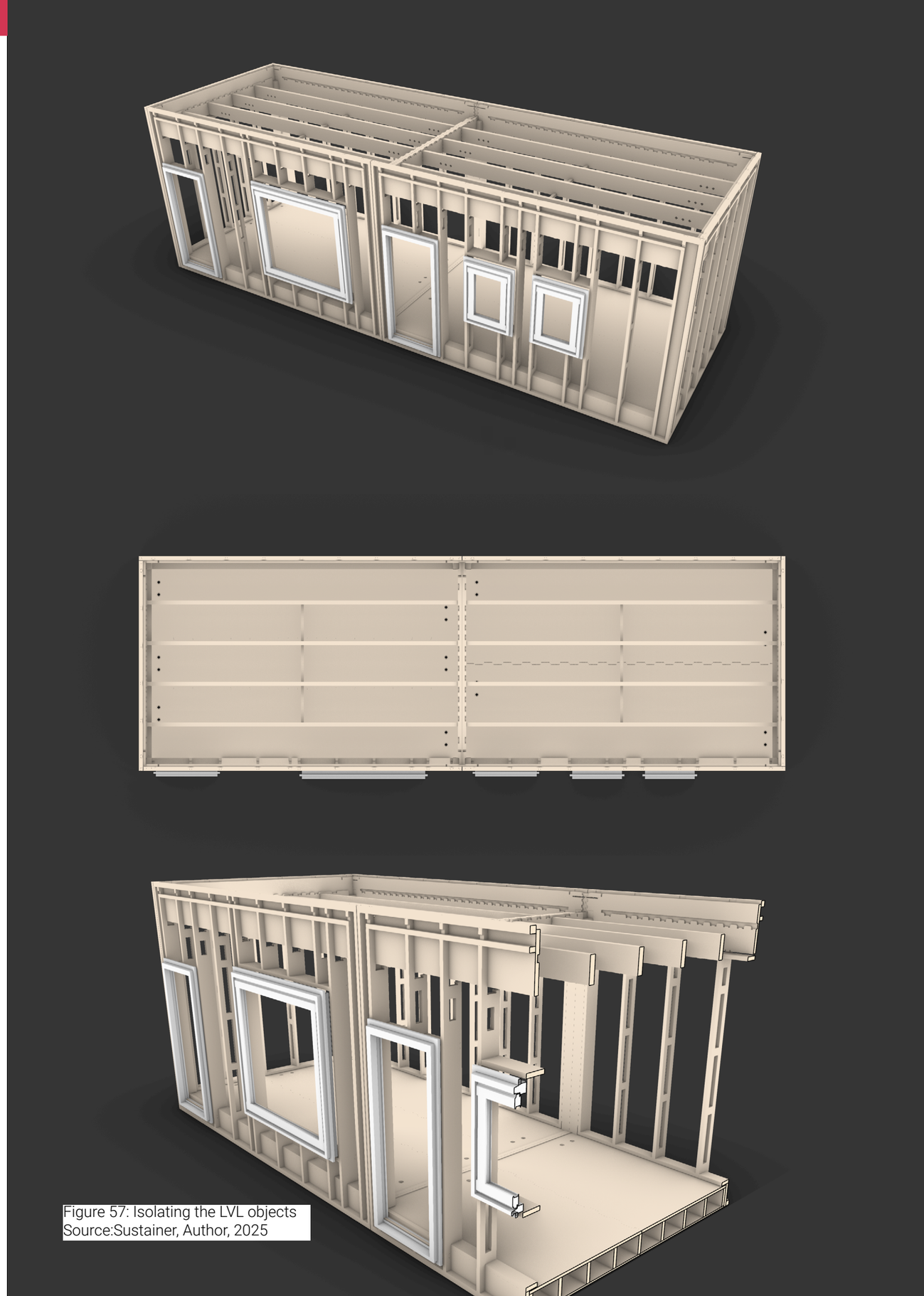


Figure 57: Isolating the LVL objects
Source:Sustainer, Author, 2025

D. From Design To Graph
D.1 A Design In The Scope Of The Research

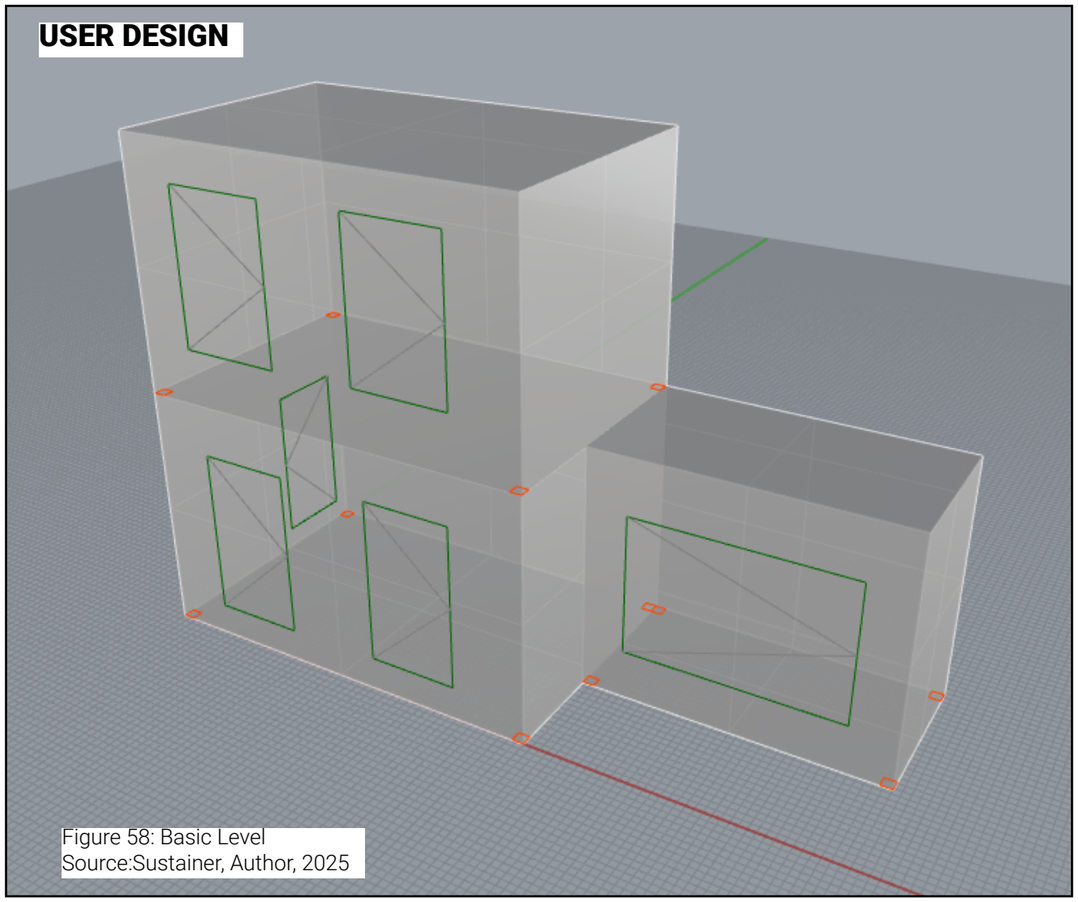


Figure 58: Basic Level
Source:Sustainer, Author, 2025

User Design Representation & GNN Input Features
The User Design is meant to be a lightweight, architect-friendly abstraction—just the 3D box placements plus their planar-curve openings—without any of the full Sustainer geometry or analysis. From this pared-down layout, we compute a set of derived features that the GNN uses to predict material volumes. Critically, the GNN never "sees" the Training Design (i.e. the detailed Sustainer meshes) at inference; it only ingests features extracted from the User Design:

1. Building Module (Node) Features
- Dimensions: length, width, height of each box
- Opening Metrics: total number of openings per module; cumulative opening area;

2.Interface & Spatial Features
- Adjacency: binary flag indicating if two modules share a face
- Interface Length/Area: length of shared edges or area of contacting faces
- Centroid Distance: Euclidean distance between module centers

3.Global Design Attributes
- Module Count: total number of boxes in the assembly
- Connectivity Metrics: list of inter-connected building modules

FROM DESIGN TO GRAPH

D. From Design To Graph
D.1 A Design In The Scope Of The Research

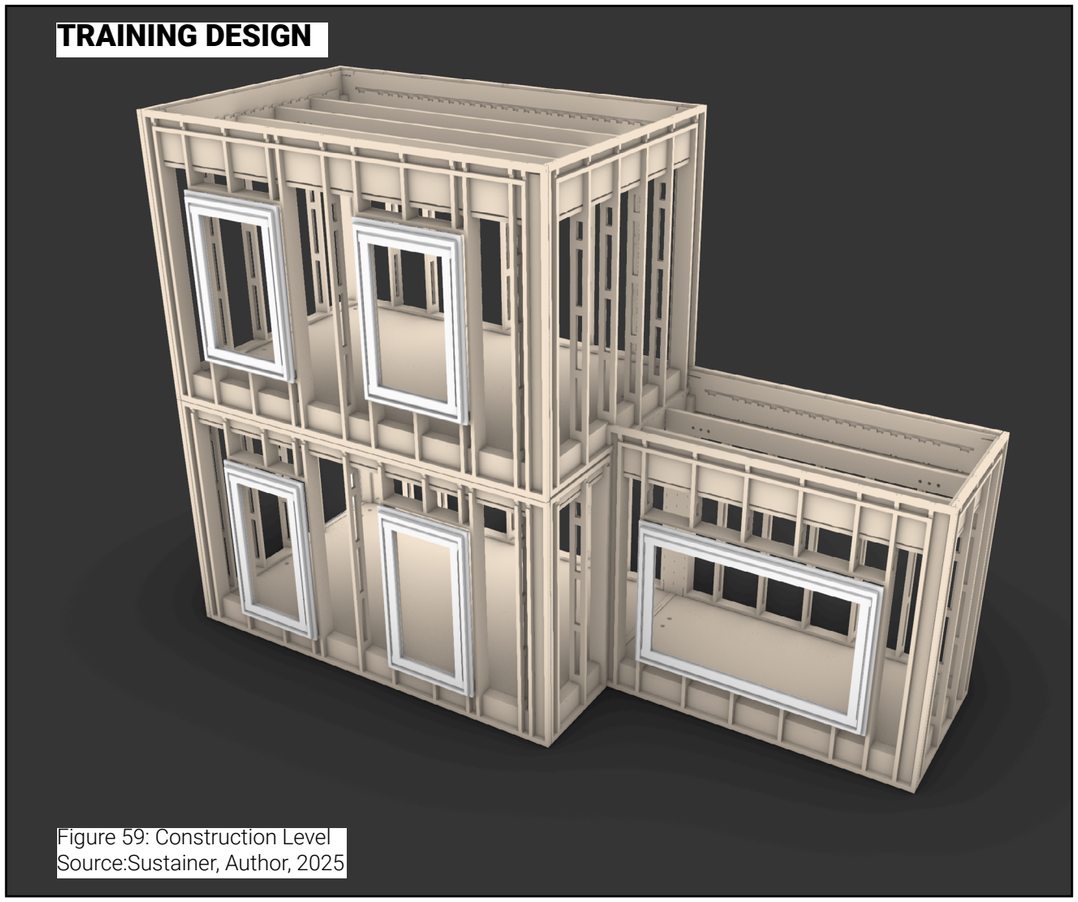


Figure 59: Construction Level
Source:Sustainer, Author, 2025

Why This Matters

No Sustainer Geometry at Inference
The GNN was trained on full Training Designs (3D boxes + curves = Sustainer's ground-truth volumes) but at runtime it only ever receives the above features—computed directly from the box + curve layout that the user sketched.

Fast, Scalable Predictions
By reducing each design to a graph of nodes (building modules) and edges (interfaces) with compact numerical features, the network can evaluate hundreds of variants in seconds.

D. From Design To Graph
D.1.1 Computational Times & The Benefits Of A GNN

Computational Demand & Scalability Challenges
Generating and processing even a simple two-module configuration through the Sustainer plugin requires 2–3 minutes of computation. As you add more modules, interfaces, and openings, this runtime grows non-linearly:

-1.5–6 modules equals to >10 minutes
-2.10+ modules equals to tens of minutes (or more)

Complex interface patterns and rich opening geometries further exacerbate processing times.

While a few minutes may seem acceptable for a one-off prototype, iterative design exploration demands dozens or hundreds of variants, each involving manufacturer choices, end-of-life/LCA scenario toggles, and structural parameters. Even with a seamless connection to Sustainer's plugin, our in-house EPD database, and bespoke Grasshopper tools, a single "what-if" branch can take hours, and a full design sweep for a large project can stretch into days or weeks.

This bottleneck makes real-time feedback impossible. It is precisely this challenge, rapidly quantifying material volume across evolving modular assemblies—**that motivates our Graph Neural Network approach. By learning to predict material volumes directly from our abstracted 3D-box and opening representations, we aim to reduce per-scenario computation from minutes to near-instantaneous, enabling truly interactive design optimization at any scale.**

FROM DESIGN TO GRAPH

D. From Design To Graph
D.1.1 Computational Times & The Benefits Of A GNN

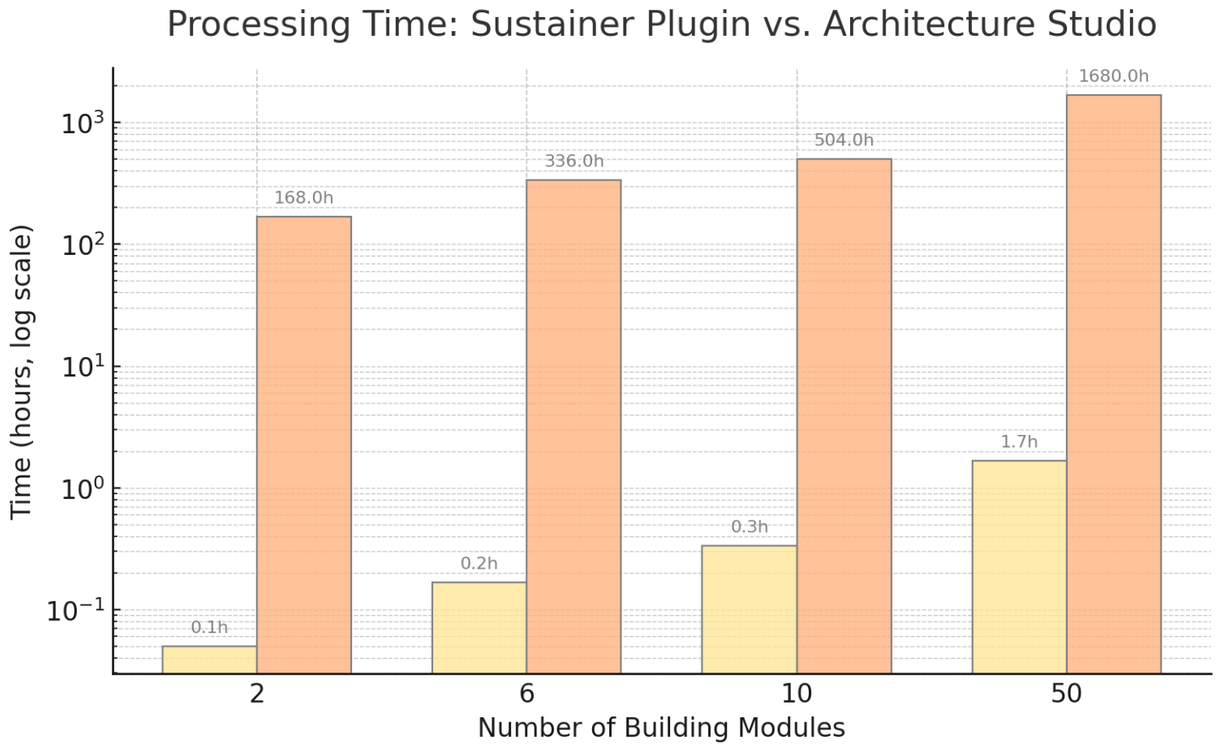
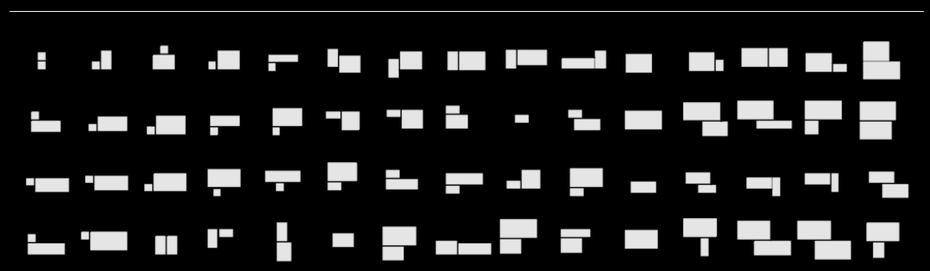


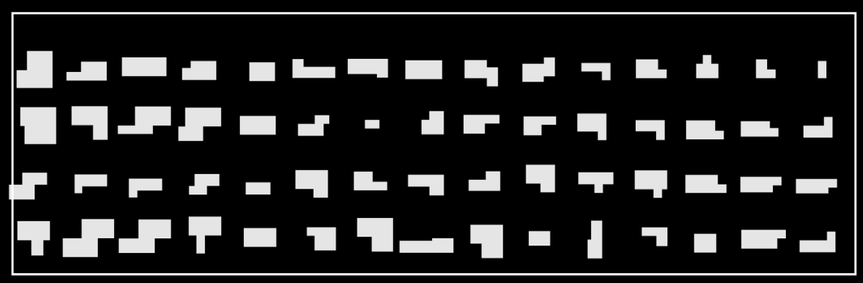
Figure 60: Computational Time
Source:Author, 2025

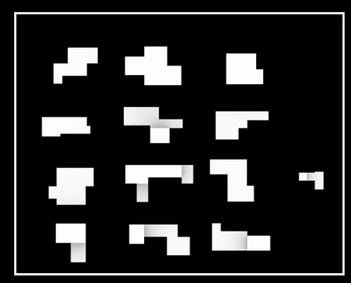
D. From Design To Graph

D.2 The Creation Of The Training Dataset

Top Views Of All The Created Training Designs

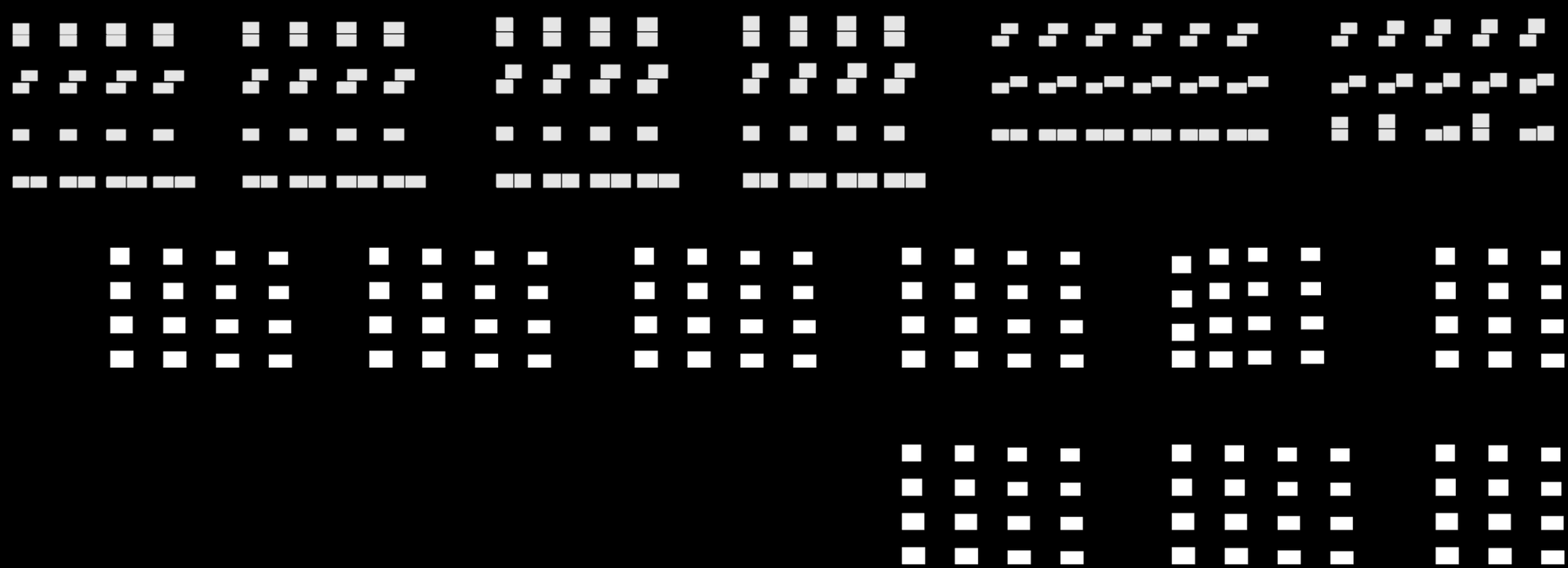


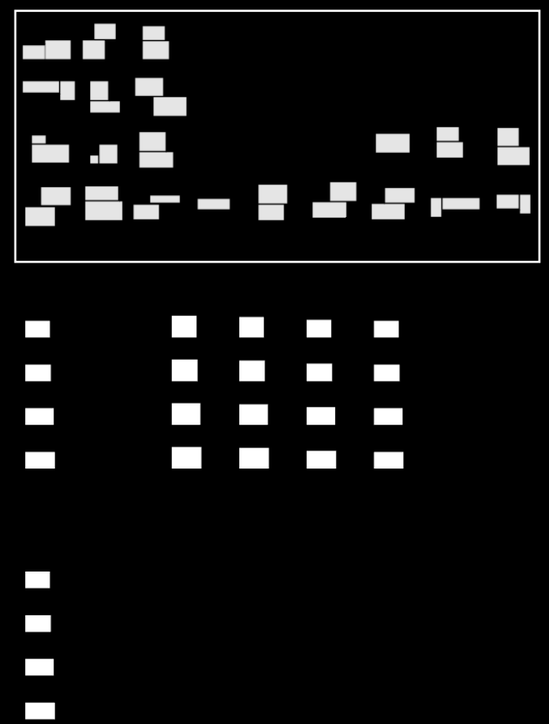




80%

Of total designs





D. From Design To Graph
D.2 The Creation Of The Training Dataset

The training dataset is organized as a collection of JSON files, each encapsulating the graph-structured features of a single design instance (which may comprise one or multiple building modules).

The following automated pipeline is employed to generate and export these files:

1. Generation of Training Designs

Batch Modeling
Multiple design variants,each defined by an arrangement of 3D boxes and their planar curve openings are authored within a single Rhino document.

Design Separation via Adjacency
An adjacency matrix is constructed by testing which boxes share faces or edges. Connected components in this matrix are identified as distinct designs, allowing multiple scenarios to be processed in one file without manual tagging.

2. Automated Feature Extraction

For each isolated design, a Grasshopper-embedded Python script performs the following steps:

Block Definition Matching
Each 3D box is linked to its corresponding Sustainer block definition, which contains the fully realized geometry and ground-truth material volumes.

Node Feature Cataloguing
The following properties are computed for every module (box):

-Dimensions: length, width, height (in meters) and discrete size keys referencing the predefined directory

-Opening Metrics: counts of doors and windows; total opening area; net face area after openings

-Intersection Areas: lateral contact areas with adjacent modules; overlap areas for vertically stacked modules

-Material Quantities: number of objects in the block definition; total LVL volume as reported by Sustainer

3.Edge & Global Attributes

-A connectivity list is generated for each node, enumerating the NodeIDs of all adjacent modules.

-Global design attributes—such as total module count and vertical stacking level—are also recorded.

D. From Design To Graph
D.2 The Creation Of The Training Dataset

Example JSON Structure Of A Design With 1 Building Module


```
{
  “design_id”: “design_001”,
  “nodes”: [
    {
      “NodeID”: “design_001_brep_0000”,
      “length_m”: 5.798,
      “width_m”: 3.358,
      “lengthkey”: 9.0,
      “widthkey”: 5.0,
      “DoorCount”: 1,
      “WindowCount”: 2,
      “OpeningArea”: 4.399,
      “AreaAfterOpenings”: 92.314,
      “SideIntersectionArea”: 0,
      “ObjectCount”: 119,
      “ObjectVolume”: 4.309,
      “connectivity”: [ ]
      “Level”: 0,
      “total_modules”: 1
    }
```


4. Dataset Scope and Diversity

Automated Batch Export
Hundreds of JSON files are produced in a single batch by iterating over all Rhino files and their contained designs.

Balanced Coverage
Module counts, size combinations, opening patterns, and stacking arrangements are systematically varied to ensure broad coverage of the design space.

Ground-Truth Integrity
Material-volume values are sourced directly from Sustainer’s block definitions, providing accurate supervision signals for GNN training.

5.JSON Structure and GNN Compatibility

Nodes & Edges: Each JSON’s nodes list and connectivity arrays map directly to GNN inputs (node-feature matrix + edge-index).

Feature Vectors: Numeric and categorical fields (dimensions, opening area, volume, size keys, level) become node features without extra parsing.

Variable Sizes: Separate files per design allow batching of graphs with differing module counts.

Decoupled Data: Topology (connectivity) and attributes (features) are clearly separated, matching standard GNN formats (e.g., PyTorch Geometric).

D. From Design To Graph
D.2.1 Latin Hypercube Sampling

But how were the training designs created?
How did we choose the layout and options each time?

Latin Hypercube Sampling (LHS) provides an efficient, stratified approach to exploring a high-dimensional design space without resorting to exhaustive enumeration. By ensuring each variable's range is uniformly partitioned and sampled exactly once per interval, LHS delivers broad coverage of all parameter combinations with far fewer samples than a full factorial design.

Methodology

1.Parameter Selection
Five key variables were chosen to characterize each module or assembly:

-LengthKey: discrete size index

-WidthKey: discrete size index

-WindowCount: integer count of window openings

-DoorCount: integer count of door openings

-ModuleCount: total number of boxes in the design

2.Interval Construction
Ech variables's continuos range min,max was divided into N equal-probability inrervals (here, N=600 to match the desried number of designs for the training dataset)

3.Stratified Sampling
A random point was drawn from each interval for every variable, then shuffled to avoid clustering across dimensions. Integer variables (counts) were rounded to the nearest whole number.

4.Scenario Allocation
To meet project-specific quotas, the 600 sampled scenarios were apportioned as follows:
-160 single-module designs (ModuleCount = 1), sampling only from 4 discrete LengthKey × WidthKey options
-200 two-module designs, same 4 × 4 sizing constraints
-150 three-module designs with identical sizing rules
-90 remaining designs with ModuleCount chosen uniformly from 1–5 and sizes drawn from the full directory

D. From Design To Graph
D.2.1 Latin Hypercube Sampling

Creating a Diverse Set of Training Designs

By combining systematic stratification with targeted quotas on module count and sizing, this LHS procedure yields a diverse ensemble of 600 training-design scenarios. It captures:

-Rare edge cases (e.g., maximal openings or unusual size pairings)

-Core configurations (single-module studies with constrained size sets)

-Intermediate assemblies (two- and three-module layouts under controlled sizing)

-Randomized multi-module mixes (up to five modules, full size directory)

This deliberate diversity ensures the GNN is exposed not only to typical design patterns but also to outlier conditions, promoting robust generalization when predicting material volumes on novel user designs.

Design Scenarios Sample Example

design_id	ModuleCount	LengthKey(s)	WidthKey(s)	WindowCount	DoorCount
design_000	1	[2.0]	[6.0]	2	1
design_001	1	[6.0]	[4.0]	0	2
design_002	1	[2.0]	[2.0]	3	0
design_003	1	[6.0]	[8.0]	1	0
design_004	1	[8.0]	[6.0]	2	1

Figure 61: Sample output by using LHS
Source:Author, 2025

D. From Design To Graph
D.3 Out Of Distribution Samples

An auxiliary dataset of out-of-distribution (OOD) designs was generated to evaluate the GNN's robustness on scenarios exhibiting substantially greater geometric and topological complexity than those in the primary training set. The OOD sampling strategy comprised the following steps:

Expanded Module Counts
Assemblies containing 6 to 7 modules (exceeding the 1–5 range of the original dataset) were created to test scalability.

Heterogeneous Size Combinations
Module dimensions were drawn from outside the four-length × four-width subset used for most designs, incorporating keys up to the full directory limits (e.g. 1.0 m–10.0 m in 0.5 m increments).

Mixed-scale assemblies (e.g. pairing very large and very small modules) ensured novel aspect-ratio distributions.

Complex Opening Patterns
Modules were assigned up to 5 windows and 3 doors, including clusters of small openings and shaped curves not seen in the original sample.

Generation Workflow

1.New Rhino files were populated with these OOD assemblies, following the same adjacency-matrix partitioning and Sustainer integration as the main dataset.

2.The Grasshopper–Python script extracted node and edge features, producing JSON files for each OOD design.

3.Dataset Composition
A total of 20 OOD designs were created, balancing extreme-scale cases (e.g. 7 modules with high opening density) and moderate-scale but topologically novel scenarios.

Purpose and Evaluation

1.Stress-Testing Generalization
The OOD dataset probes the GNN's ability to extrapolate beyond its training distribution, identifying failure modes on unusually large, irregular, or intricately connected assemblies.

2.Performance Metrics
By comparing prediction errors on the OOD set against in-distribution test cases, it becomes possible to quantify how well the model captures fundamental volume-mapping principles versus overfitting to the original design patterns.

This OOD evaluation framework ensures that the GNN's deployment readiness is measured not only on typical modular configurations but also under demanding, previously unseen conditions.

D. From Design To Graph
D.4 Conclusions

In this chapter the pipeline for transforming architectural modules into graph-structured data suitable for Graph Neural Network training is showcased. Key achievements include:

1.Design Taxonomy
The clear separation between Training Designs (full Sustainer geometry + ground-truth volumes) and User Designs (abstract box+curve layouts) ensures that the GNN learns a mapping from lightweight inputs to accurate volume estimates without reliance on detailed meshes at inference.

2.Automated Dataset Generation
A Grasshopper-embedded Python workflow was developed to batch-process Rhino files, partition multiple scenarios via adjacency matrices, and extract rich node and edge features. Each design is serialized as a standalone JSON, directly matching the node-feature/edge-index conventions of modern GNN libraries.

3.Strategic Sampling
Latin Hypercube Sampling produced 600 training cases with controlled variation in module count, size, and opening patterns—balancing core, intermediate, and randomized configurations.

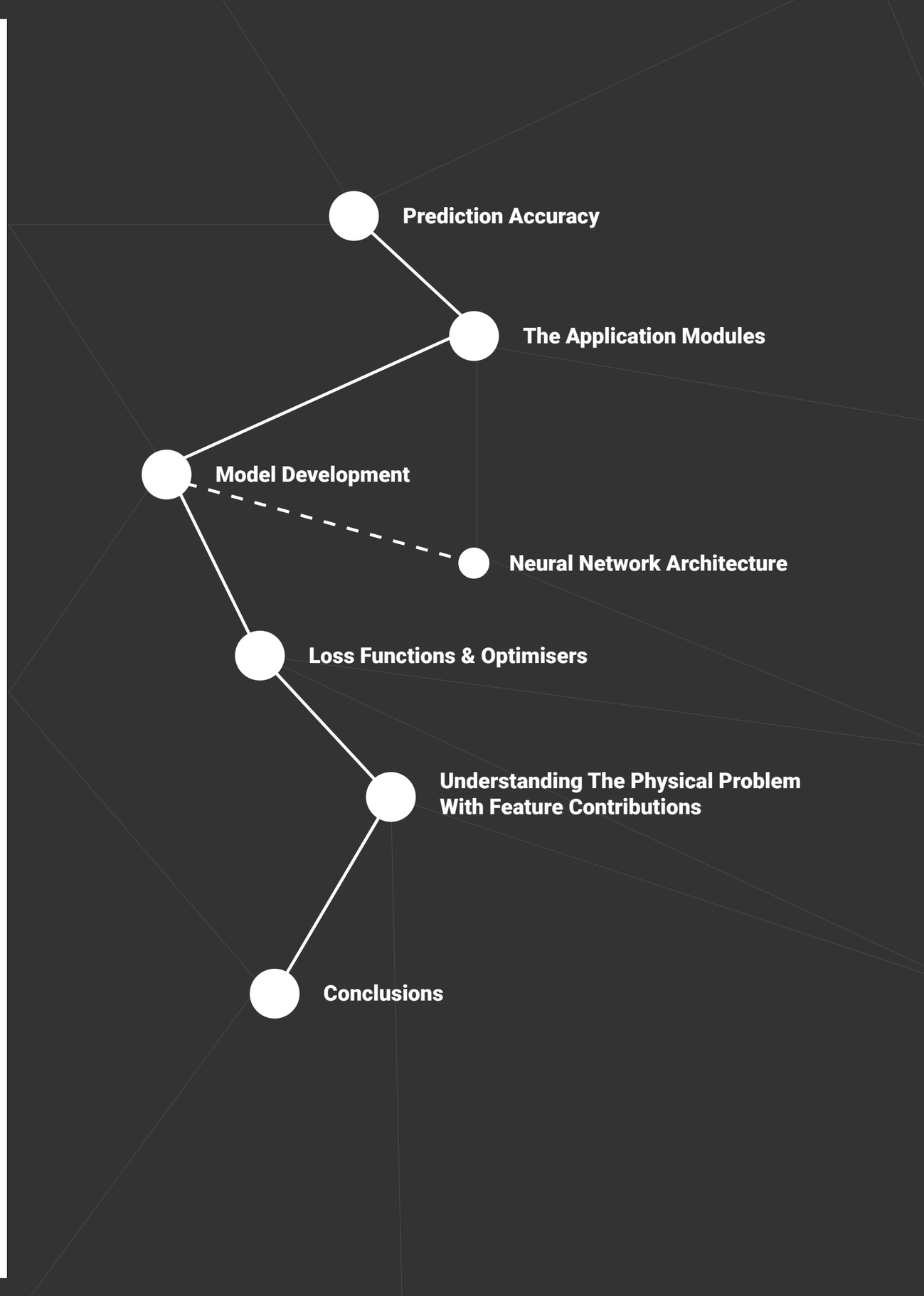
4.Out-of-Distribution (OOD)
Samples introduced higher module counts, novel size combinations, complex openings, and irregular connectivity to stress-test model generalization.

5.Graph Compatibility
The JSON schema from attributes (dimensions, opening metrics, volumes) streamlines data loading and batching, reducing pre-processing overhead and facilitating integration with frameworks such as PyTorch Geometric or DGL.

Collectively, these components yield a diverse, scalable, and machine-readable training corpus that equips the GNN to predict material volumes across both familiar and novel modular configurations. The methodology ensures rapid data generation, robust generalization testing, and a clear pathway

End Of Chapter D. From Design To Graph

Work Package 03 | Graph Neural Network Training & Evaluation



E. A Graph Neural Network For Predicting Material Volume

Brief Summary

This chapter develops a Graph Neural Network (GNN) surrogate for rapid, module-level LVL volume—and by extension embodied-carbon and other predictions. Initial tests (E.1) revealed under-fitting and poor generalization (~80 % in-distribution, ~70 % OOD). A reproducible six-module codebase (E.2) orchestrates data loading, graph construction, model definition, training, evaluation, and visualization.

Architectural refinements (E.3) introduced a pre-embedding MLP, three graph-attention layers with normalization, residual connections, dropout, AdamW + Huber loss, and learning-rate scheduling. Finally, a monotonic-decreasing skip branch and sign-regularization penalty enforce that door, window, and intersection features always subtract volume. These changes stabilize training, smooth accuracy curves, and boost validation performance into the mid-80 % range.

Feature-attribution via Integrated Gradients (E.4) confirmed that only the constrained model correctly assigns negative contributions to void features. The result is a transparent, high-fidelity surrogate that—when paired with EPD data—delivers immediate, interpretable life-cycle feedback during concept design, bridging the gap between circular-economy goals and early-stage practice.

E.A Graph Neural Network For Predicting Material Volume

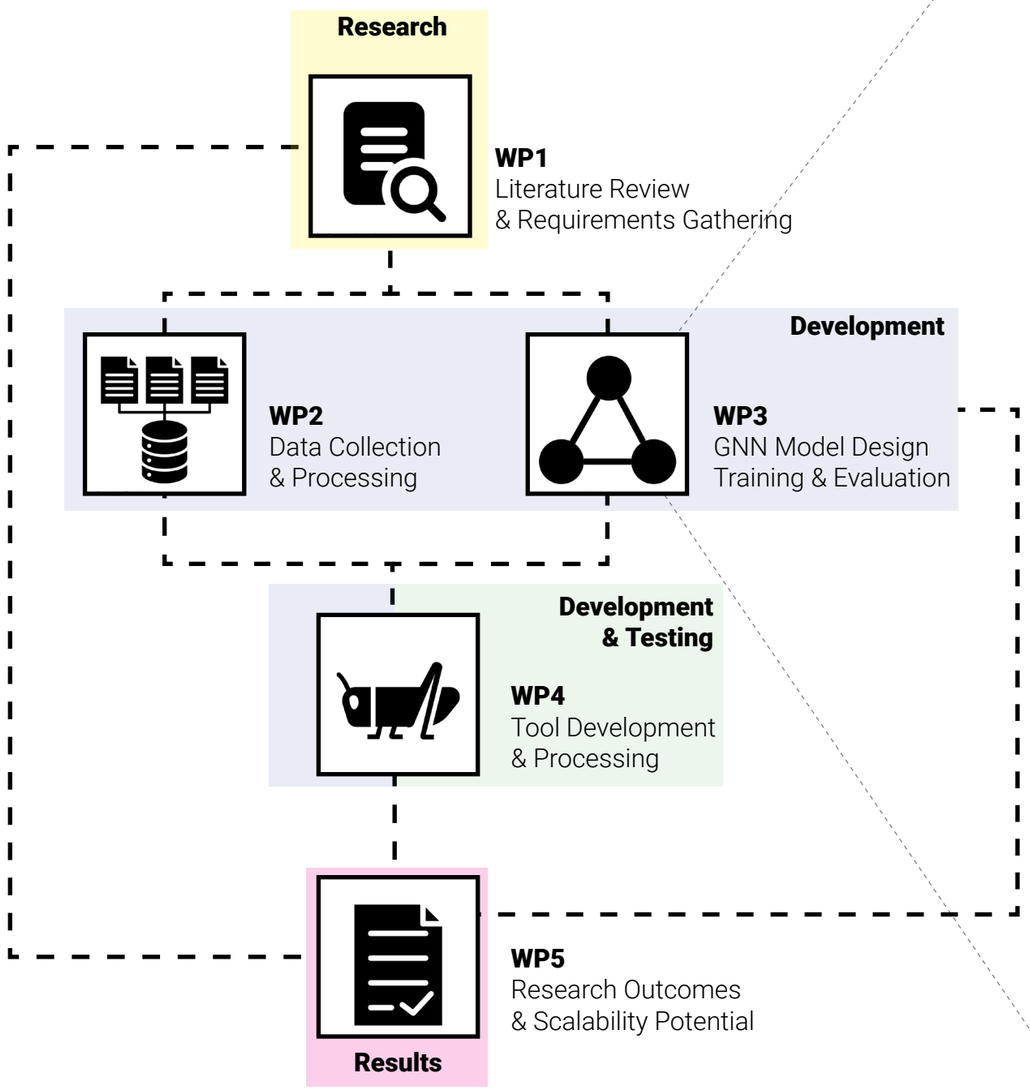


Figure 11: Work Packages Overview / Source: Author, 2025.

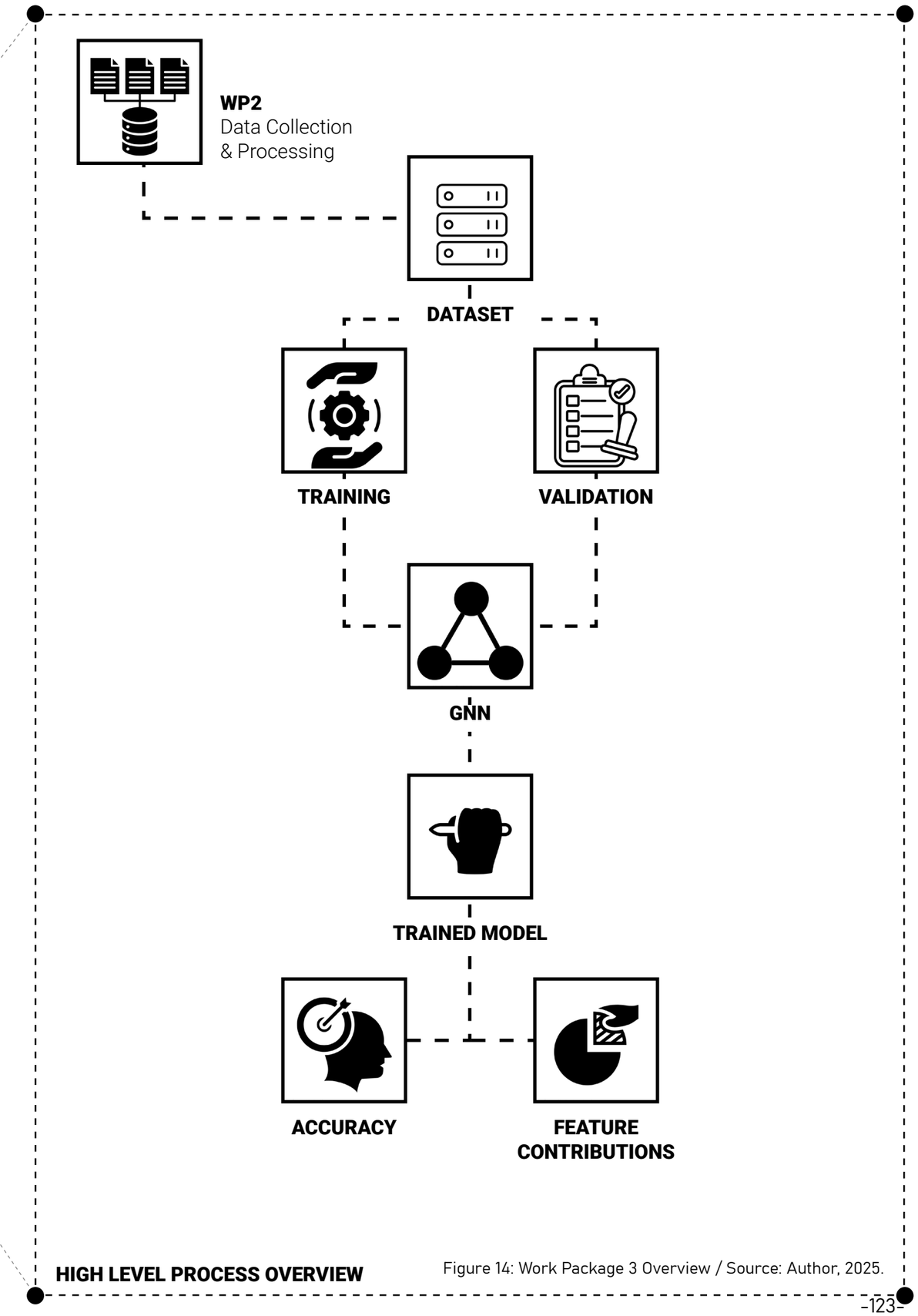


Figure 14: Work Package 3 Overview / Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.1 Prediction Accuracy

Having assembled an EPD database that consolidates per-cubic-meter LVL metrics from four leading European manufacturers, and having generated the comprehensive, graph-structured training corpus detailed in Chapter D, the focus now shifts to the Graph Neural Network itself. This core component ingests only early-stage design abstractions, 3D module boxes, planar-curve openings, and their connectivity and produces near-instantaneous material volume predictions.

By learning the intricate relationships between module dimensions, opening characteristics, and inter-module interfaces, the GNN obviates time-consuming geometry processing and delivers real-time feedback to support rapid design iteration.

Prediction accuracy must be evaluated on two levels.

1.Describe the numerical fidelity of the GNN's outputs, how closely the predicted LVL volumes and component counts match the ground-truth values extracted from Sustainer. This is quantified via mean absolute percentage error.
For example: A high numerical accuracy ensures that, when the network predicts 4.2 m³ of LVL for a given layout, the true volume will fall within a tight tolerance, minimizing the risk of under- or over-specification.

2. Accuracy extends beyond raw volume estimates. The predicted volumes feed directly into life-cycle and environmental assessments via our EPD database. When the model outputs a per-module volume, that figure is multiplied by manufacturer-specific impact factors and aggregated to yield the final environmental metrics. In this context, "accuracy" also implies that the combined pipeline (GNN prediction plus EPD lookup) produces credible carbon footprint or material-cost forecasts that designers and stakeholders can trust.

Example:
The selected LCA scenario is Cradle To Grave & Beyond (A1-D).
The selected EOL scenario is Reuse.
Manufacturer: Stora Enso

From the EPD database (kg CO2 eq.):
A1: − 693 | A2: 18.8 | A3: 27.0 A4: 27.3 | A5: 6.08
B1–B7: 0 | C1: 4.01 | C2: 2.22 | C3: 968 | C4: 0 |D: − 331

Summing A1 through D gives the cradle-to-grave-and-beyond factor:

$$F_{\text{net}} = (-693) + 18.8 + 27.0 + 27.3 + 6.08 + 0 + 4.01 + 2.22 + 968 + 0 - 331 = 29.41\,\text{kg CO}_2\text{ eq / m}^3$$

With a true module volume of 4.80 m³, the actual embodied carbon is

$$V_{\text{true}} = 4.80\,\text{m}^3, C_{\text{true}} = V_{\text{true}} \times F_{\text{net}} = 4.80 \times 29.41 = 141.17\,\text{kg CO}_2\text{ eq}$$

E.A Graph Neural Network For Predicting Material Volume

E.1 Prediction Accuracy

An 85 % volume-prediction accuracy yields

$$\hat{\text{HV}} = 4.80 \times 0.85 = 4.08\,\text{m}^3$$

$$\hat{\text{HC}} = \hat{\text{HV}} \times F_{\text{net}} = 4.08 \times 29.41 = 119.99\,\text{kg CO}_2\text{ eq}$$

The 15 % under-prediction (0.72 m³) thus causes a 21.18 kg CO2 eq shortfall, also 15 %, in the environmental forecast, demonstrating how GNN volume-prediction error propagates through the EPD lookup to the final carbon metric.

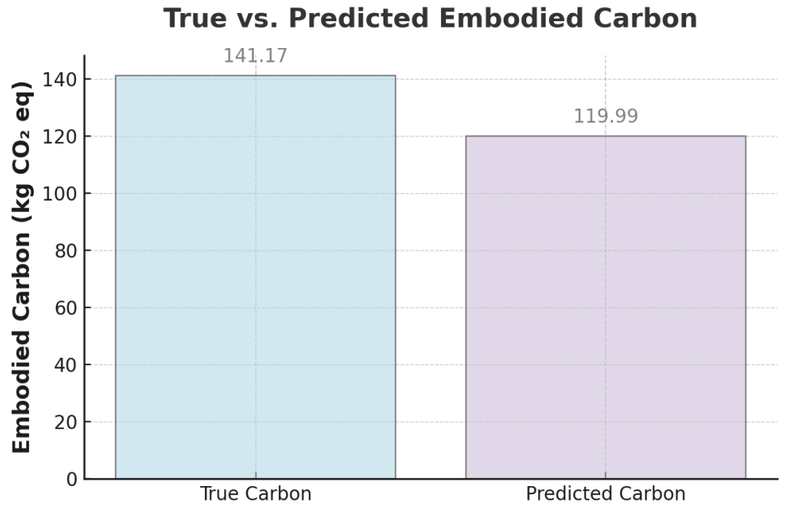


Figure 62: Actual Vs Prediction / Source: Author, 2025.

In the surrogate-modeling literature, an 85 % predictive accuracy (i.e., a 15 % MAPE) occupies a middle ground between rapid "screening-level" estimates and high-fidelity requirements:

First, Song et al. (2018) showed that artificial-neural-network surrogates for life-cycle impact assessment can achieve R^2 values around 0.48–0.87 depending on the impact category, with MAPE often exceeding 15 % for global-warming potential—indicating that 15 % error is common but leaves substantial room for improvement (Song et al., 2018, https://doi.org/10.1021/acs.est.7b02862).

Second, in the domain of building-energy prediction, Liu et al. (2023) report that surrogate models typically require MAPE below 10 % (equivalent to > 90 % accuracy) before being considered reliable for design decisions, with anything above this threshold deemed only "preliminary" (Liu et al., 2023, https://doi.org/10.3390/buildings15081301).

Taken together, an 85 % accuracy can suffice for early-stage exploration, where speed outweighs precision, but it falls short of the > 90 % benchmarks that practitioners expect for final embodied-carbon or cost analyses. To bridge this gap, further model refinement or hybrid workflows—combining fast GNN estimates with selective detailed validation, can help ensure both rapid feedback and decision-grade reliability.

E.A Graph Neural Network For Predicting Material Volume

E.2 The Application Modules

So how is the core component that provides the predictions is structured?

Through the project's timeline and especially during the development of WP3, the GNN was rigorously evaluate and tested to meet its main 2 goals:

1) Provide material volume predictions with an accuracy level above 85%
2) Evaluate if the GNN understands the physical problem by assessing the feature contributions.

But what is the physical problem in our case?

The physical problem associated with the matter at hand is to see if the model understands how the user's input increases or decreases the material volume. For example, when we apply an opening to the building module, the logical reaction is that the material volume will decrease because of the void necessary but if the feature attribution will show the a larger area of openings increases the predicted value, even if we achieve a high level of accuracy that means that the GNN does not understand the physical relationship of the features.

The Graph Neural Network was constructed to transform the JSON files produced from the training designs (Chapter D) to graphs which the GNN will use to try to learn how the features result to the predicted value and get accurate module-level volume estimates, balancing flexibility, interpretability and performance. The development unfolded in several stages, each chosen for clear rationale.

The application is also seperated into modules for better handling.

1.Configuration

Here are the most important settings:

a)File paths : The locations of necessary folders are provided like where is the training dataset location, or to which folder the trained model should be saved.

b)Toggles: These act "controls" which influence how the application will operate.

-Train Mode / Evaluation Mode : This is a really important addidtion since after initial training, in most cases we do not want to retrain the model but to produde only visualisations for evaluation puroposes.

-Plot/Display Visualisations: Instead of saving the visualisation images each time we run the model, the application has the option to just preview them instead of saving them and increasing allocated space.

c)Logging: Control the amount of information that is printed while running for debug purposes.

d)Hyperparameters: These are the core controls that set the training process of the neural network.

-Batch size: A integer number that sets how many JSON files the neural network will process in each batch at a time during training (Bottou, L. ,2010)

-Epochs: An epoch is one complete forward and backward pass of all the training examples in the dataset (Goodfellow et al., 2016; DeepAI, 2017)

-Learning Rate: A hyperparameter in an optimization routine that controls the magnitude of each update as the algorithm iteratively converges on the loss function's lowest point. (Robbins & Monro, 1951; Ruder, 2016)

-Dataset Split: A fraction sets how many of the json files will be allocated for the training of the neural network and how many for its validation.

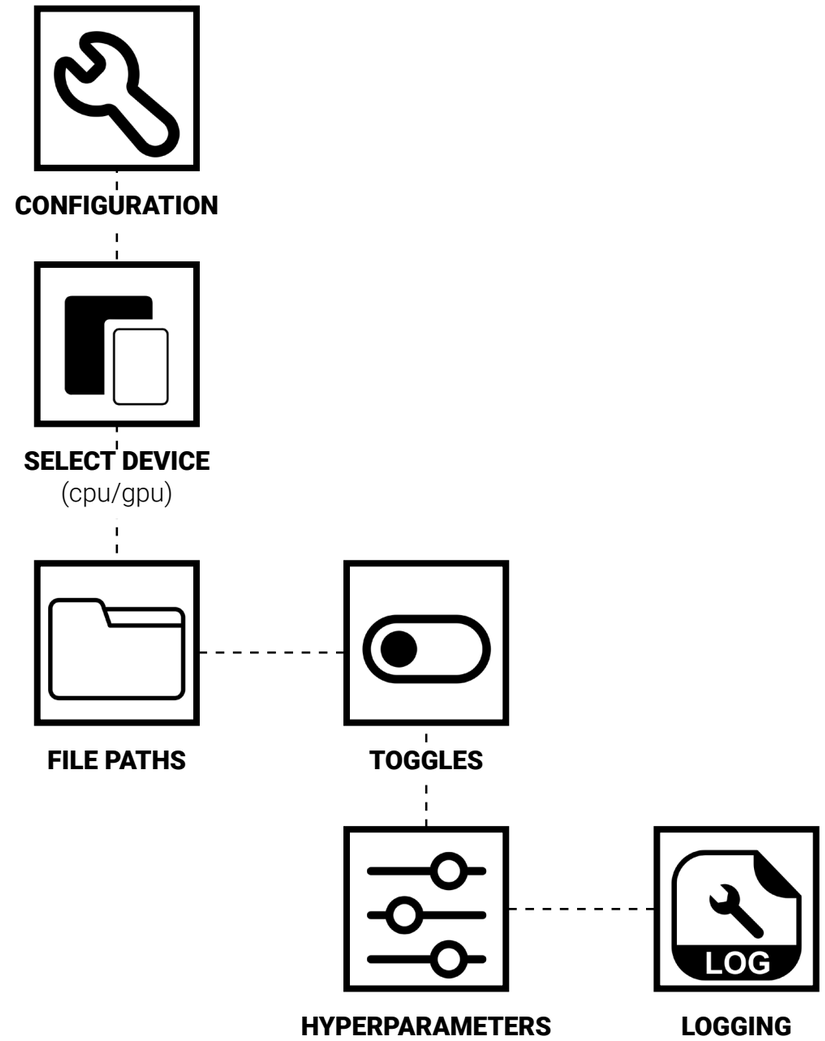


Figure 63: Overview of the Configuration Module / Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.2 The Application Modules

Then either if the model will execute or will skip it to proceed directly to evaluation, the next step is to access the module responsible for the dataset.

Dataset Module

The dataset module handles the creation of the training and validation dataset from the directory of the json files given by the configuration module. Afterwards:

1. The application **loads the json files**, logs any errors through the configuration module if any.

2. From each file **a graph is created in the form of tensors.**

3.Through the hyperparametes from the configuration module **the graphs are split, some for training and some for validation.**

4.Custom bather: Here **the process of loading and validating the files is isolated** so we can find out if there is any problematic graph.

The Graph Regressor Module

A graph regressor is responsible for mapping a graph-structured input to continuous target values on its nodes. In general, it operates through the following stages (see Battaglia et al., 2018; Gilmer et al., 2017; Kipf & Welling, 2017):

1.Node Embedding
Every node's raw feature vector (e.g. geometry, opening metrics, module count) is projected into a latent space by a learnable function. This brings heterogeneous inputs onto a common scale and allows the model to discover useful feature combinations (Battaglia et al., 2018, arXiv:1806.01261).

2.Message Passing / Neighborhood Aggregation
Over multiple iterations, each node gathers "messages" from its neighbors—transformed versions of their embeddings—and integrates them into its own representation. This process captures spatial context: how adjacent or connected modules influence one another's material requirements. Architecturally, this can take the form of spectral graph convolutions (Kipf & Welling, 2017, arXiv:1609.02907) or more general message-passing layers (Gilmer et al., 2017, arXiv:1704.01212).

3.Residual or Skip Connections
To prevent important local information from being washed out by repeated aggregation, the original node embedding is often carried forward in parallel (a "skip"), then merged with the final aggregated vector. This ensures that extreme feature values—such as a very large opening—remain visible to the readout stage (Kipf & Welling, 2017).

E.A Graph Neural Network For Predicting Material Volume

E.2 The Application Modules

4.Readout / Regression Head
Once a node's final representation encodes both its own attributes and the context of its neighbors, a small feed-forward network maps that vector to one or more continuous outputs (e.g. volume, count). Jointly learning multiple targets can improve data efficiency by exploiting correlations between tasks (Gilmer et al., 2017).

5.Regularization
Techniques such as dropout on hidden activations or weight-decay on parameters guard against overfitting, encouraging the model to learn broadly applicable patterns rather than memorizing specific graph instances (Wu et al., 2021, DOI:10.1109/TNNLS.2020.2978386).

During training, the model's parameters are optimized to minimize a regression loss between its predictions and known ground-truth labels from the validation that was creatted previously. By combining these components, a graph regressor generalizes the convolutional paradigm to irregular, connected data structures, making it uniquely well-suited to predict continuous material metrics on assemblies of building modules.

The Utilities Module

A utilities module is the quiet infrastructure that makes an experimental code-base behave like a well-run laboratory of four broad service areas:

1.Reproducibility
The module keeps the workspace orderly. Because each experiment leaves a clean "paper-trail," you can reliably revisit past results or audit how old results were produced.

2.Data–Model Glue
The utilities layer standardises that journey and gathers model outputs in neatly structured arrays.

3.Quantitative Evaluation
Progress only becomes meaningful when measured the same way every time. A utilities module therefore supplies a single source of truth for metrics—error rates, accuracy, and any project-specific scores and prints friendly summaries or alerts when something drifts.

4.Visualisation & Interpretability
Graphs and heat-maps turn raw numbers into insight. The utilities layer handles that transformation end-to-end: it applies a consistent visual style, decides whether plots are shown interactively or archived, and most importantly, uses those visuals to critique the model itself. Performance dashboards—loss curves, prediction-vs-truth scatters, validation trend lines—let you spot overfitting, convergence stalls, or sudden metric spikes at a glance.

Feature-attribution plots for example, heat-maps based on integrated gradients or similar techniques reveal which design properties the model relies on. When those highlights correspond to physically meaningful features, you gain confidence that the network is learning the right story.

E.A Graph Neural Network For Predicting Material Volume

E.2 The Application Modules

In total the Graph Neural Network (GNN) is made up by 6 modules:

1. Configuration Module
Handles most of the applications 's main settings, file paths and controls

2.Main Module
Directs the order of actions and exceptions in the application.

3.Model Module
Contains the architecture of neural network.

4.Training Module
Handles training, optimisation and loss functions.

5.Dataset Module
Creates the graphs from the JSON files.

6.Utilities Module
Holds all the methods for debugging, plotting and visualisation methods of the GNN.

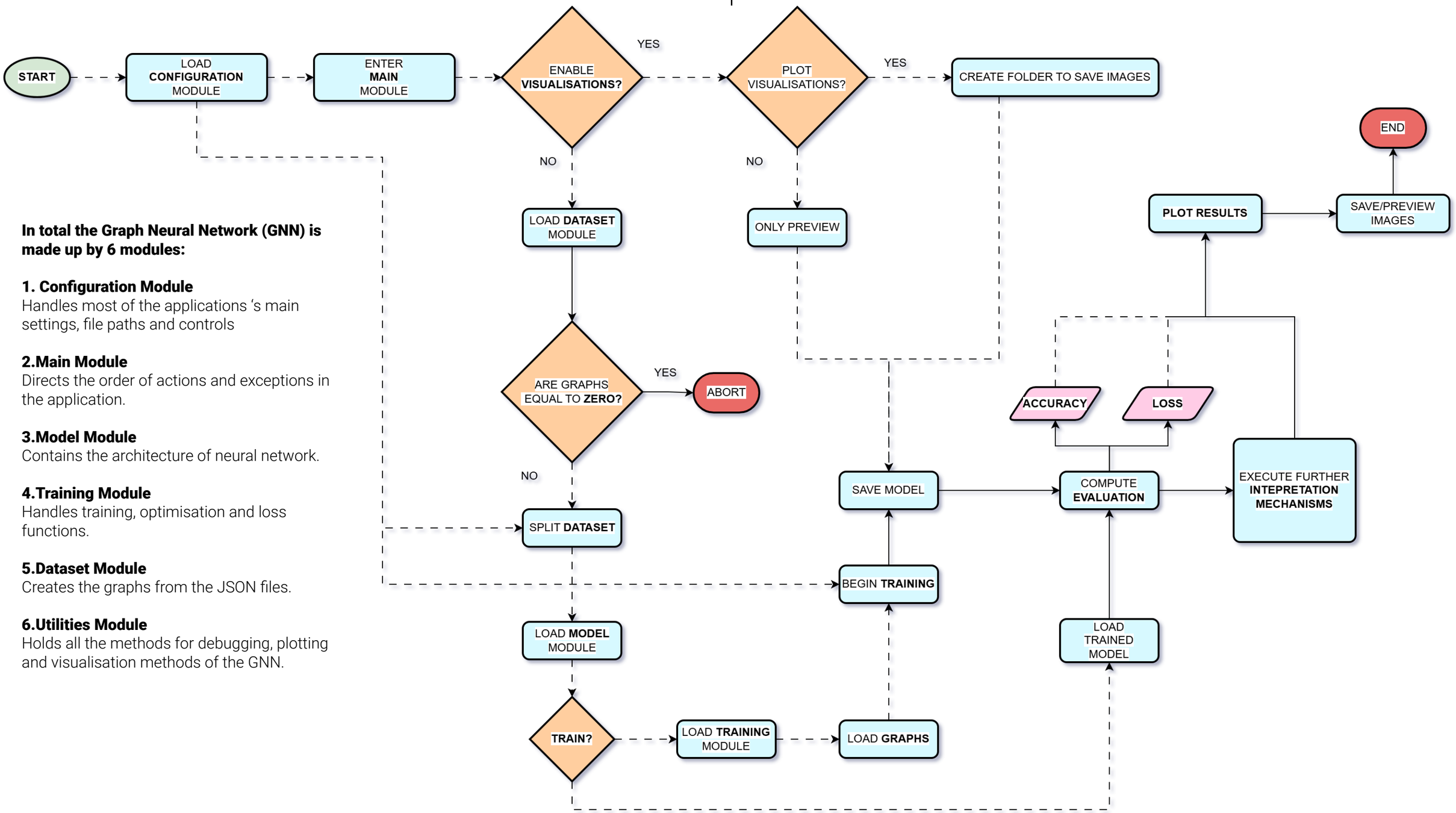


Figure 64: GNN order of actions/ Source: Author, 2025.

GRAPH NEURAL NETWORK DEVELOPMENT

E.A Graph Neural Network For Predicting Material Volume

E.3 Model Development

With the data pipelines, training logic and utility scaffolding firmly in place, we can now turn the spotlight onto the neural network itself, examining, layer by layer, how its architecture was engineered to deliver dependable material-volume estimates for every building module.

The next section will dissect the choice of the architecture and type of GNN and the rationale behind node-wise normalisation and other methods that keep the model's physics honest. Along the way we will revisit the practical hurdles that surfaced and how they were approached.

E.3.1 Neural Network Architecture

First Tests

A three-layer message-passing network lies at the heart of the volume estimator. Each module in the design is represented as a node carrying its measured dimensions (length, width, height), the number and total area of its openings, and other simple statistics. Whenever two modules share a face or an edge, a connection is drawn between their nodes to reflect that spatial relationship.

The first processing layer takes each node's raw measurements and converts them into a 32-component internal description. **In effect, it learns which geometric features, say, a particularly large window or a tall wall, matter most when estimating material. A basic non-linear activation follows, enabling the model to capture more than just straight-line relationships.**

The second and third layers perform further rounds of communication: each node gathers information from its neighbors, combines it with its own data, and passes the result onward. **Including each node's own data at every step (a "self-connection" the green ribbon at figure 71) ensures that original measurements never disappear entirely, while a simple normalization prevents heavily connected modules from overwhelming the group's shared information.** By the end of these three passes, each module's internal description reflects both its own geometry and the broader context of the entire assembly.

A final, purely linear mapping then translates each 32-component description into two numbers, the predicted count of LVL objects and the total LVL volume, without any extra non-linear tricks. This choice forces the earlier layers to organize their internal representations around genuinely volume-relevant factors, rather than hiding complexity in the final step.

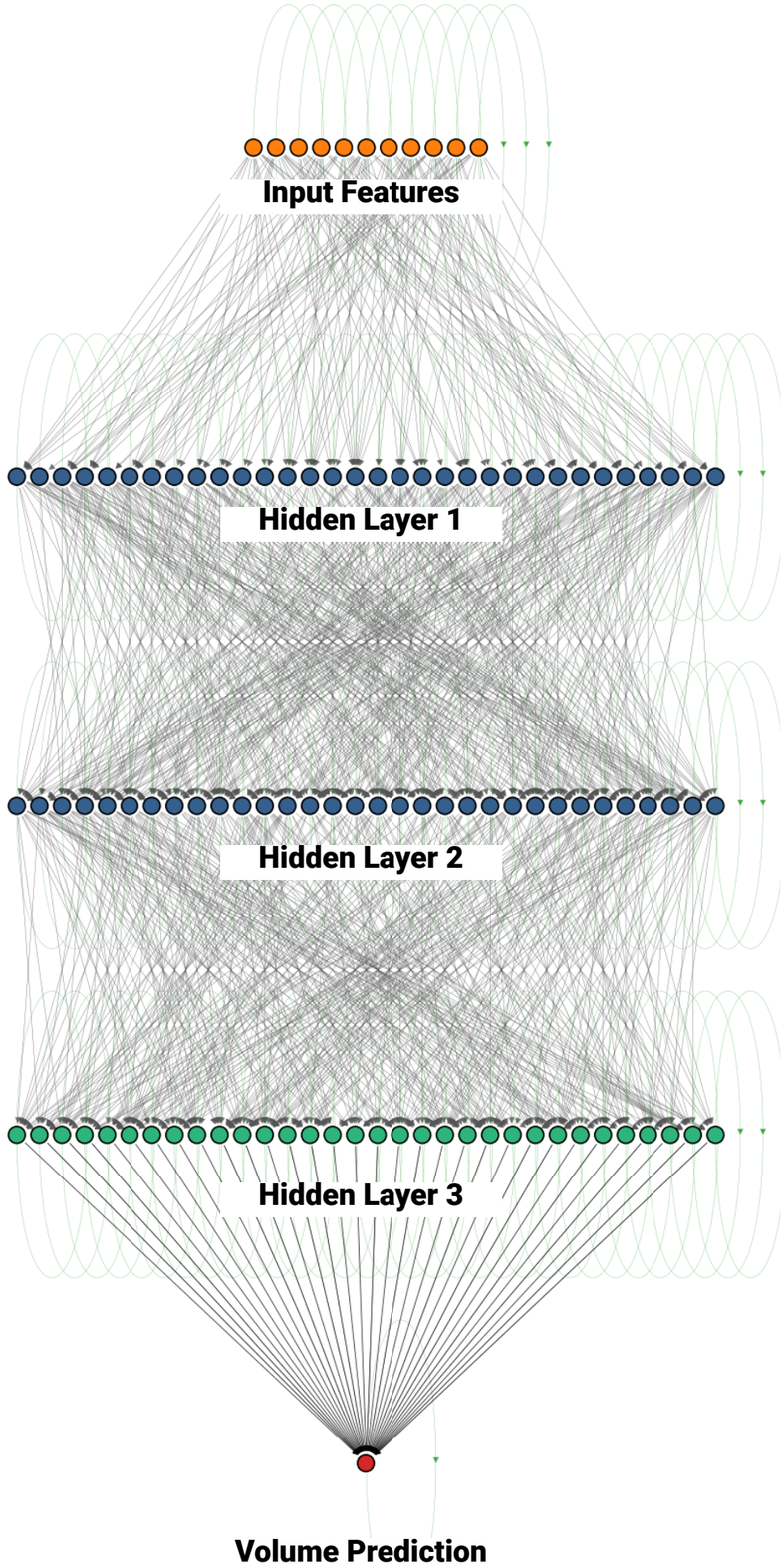


Figure 65: First GNN model with 3 convolutional layers and self loops/ Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.3.1 Neural Network Architecture

Here are some short definitions for better understanding of the results in the following chapters:

a)Training loss measures the error on the data used to update the model's parameters (Deep Learning, Ian Goodfellow et al., 2016, ISBN 978-0262035613).

b)Validation loss evaluates error on a separate held-out dataset to monitor generalisation (Deep Learning, Ian Goodfellow et al., 2016, ISBN 978-0262035613).

c)Convergence refers to the point at which additional training yields negligible improvement (Deep Learning, Ian Goodfellow et al., 2016, ISBN 978-0262035613).

d)Under-fitting occurs when a model is too simple to learn the data's complexity, leading to a performance plateau (The Elements of Statistical Learning, Hastie et al., 2009, ISBN 978-0387848570).

e)Over-fitting occurs when the model learns noise or idiosyncrasies from the training data, achieving low training loss but poor validation performance. It reflects a loss of generalization as the network tailors itself too closely to the training examples (The Elements of Statistical Learning, Hastie et al., 2009, ISBN 978-0387848570).

f)Out-of-distribution (OOD) samples are test cases whose characteristics lie outside the range seen during training ("Domain Generalization: A Survey," Zhou et al., IEEE Trans. Pattern Anal. Mach. Intell., 2023, DOI 10.1109/TPAMI.2022.3195549).

Results

1. Although training loss and validation loss remain closely aligned,showing that the model does not memorise the training data at the expense of generalisation, the network reaches its best performance very quickly and then levels off, failing to improve further with more training. This plateau in learning indicates under-fitting, meaning the model is too simple to capture the underlying patterns fully. Accuracy on in-distribution data never climbs above 80% and when the model is tested on out-of-distribution samples, designs that differ markedly from the training set, its accuracy drops to around 70%. **Such performance shortfalls are unacceptable for the goals of this research.**

2. Although each node includes a self-connection (an edge from the node back to itself) to preserve its individual feature information during message passing, this safeguard proves insufficient. On out-of-distribution designs, the model assigns the same predicted volume to every module, even though the modules differ in size and opening configuration. Verification confirmed that the input graphs did not contain identical modules, so the uniform predictions cannot be explained by identical inputs. This behavior suggests that the self-connection mechanism alone fails to enforce distinctive representations for each node under novel conditions.

Example in one OOD design:
Node 01 : ObjectVolume=3.5224
Node 02 : ObjectVolume=3.5224

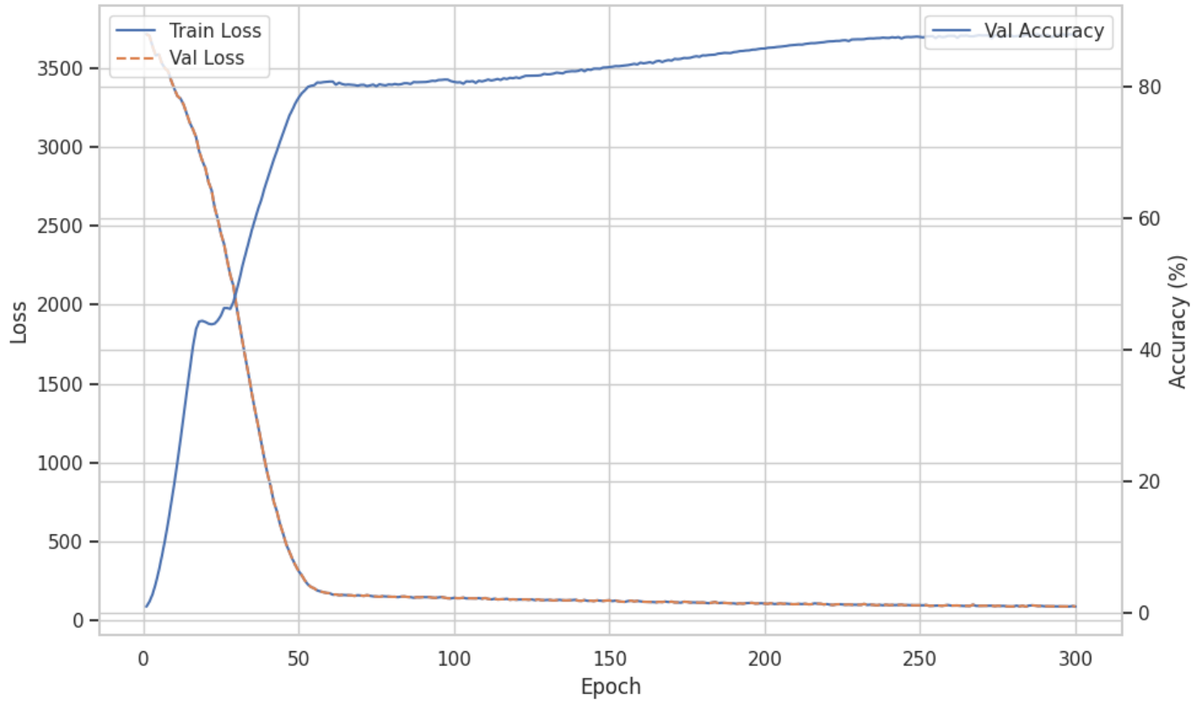


Figure 66: Accuracy levels flatten over time / Source: Author, 2025.

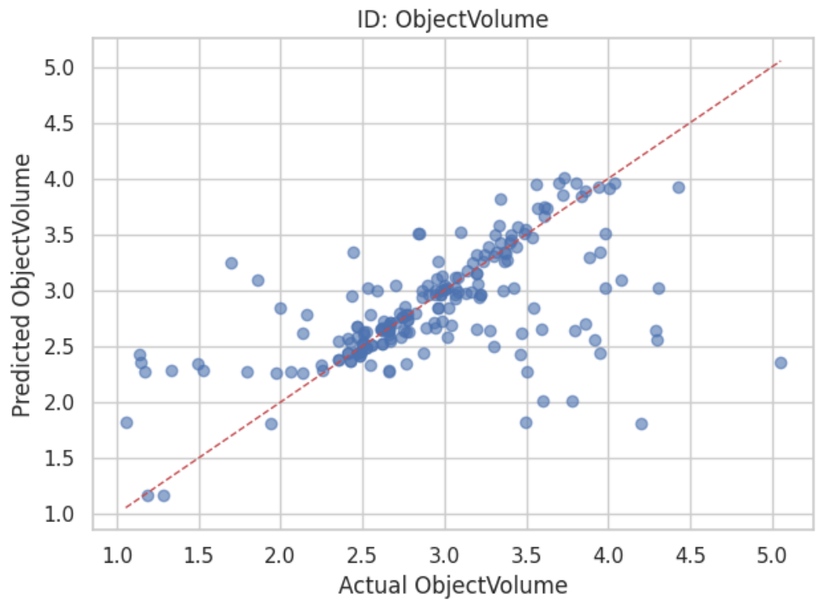


Figure 67: With a level of accuracy near 80%, still many building modules are not accurately predicted when it coems to volume. / Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.3.1 Neural Network Architecture

-Solving the identical predictions across all the participating building modules in a design.

Instead of letting the three rounds of neighborhood averaging wash out each node's unique traits, **a simple "shortcut" was addied that carries a node's original measurements (length, width, openings, etc.) straight into the final layer.** Concretely, each node's raw feature vector is taken and run through a small linear layer to match the hidden size, and then it is added to the output of the third layer.

This bypass means a node never loses its own information entirely, so even after blending in its neighbors' data, it still "remembers" its own dimensions. The result is that modules of different shapes or opening sizes once again get distinct volume predictions, rather than collapsing to the same value, even on designs the network hasn't seen before.

New Results

1. While finally we get unique predictions per building module, **a problem still persists a drop of 5% to 7% in accuracy when the model was introduced to the OOD samples.**

2.This was probably due to the fact that the model still converged rather quickly, **which hinted that it did not still fully grasp the physical relationships of the input features.**

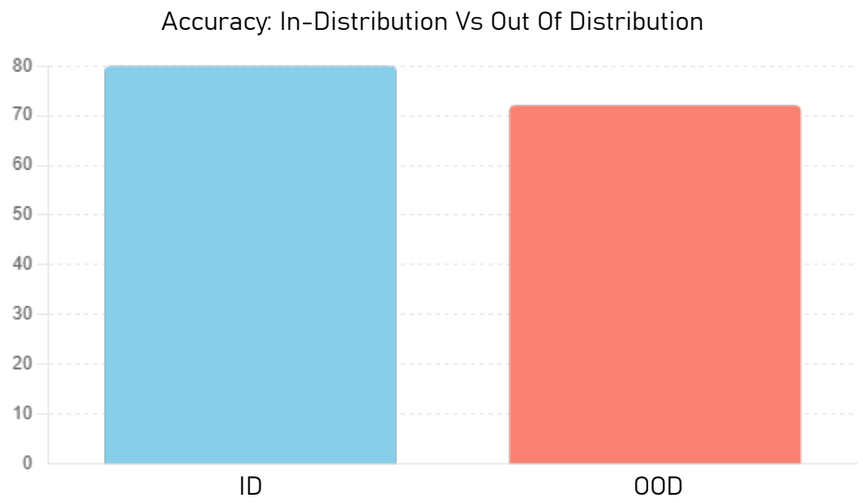


Figure 68: Accuracy drops when the model sees a design that does not exist in the dataset
Source: Author, 2025.

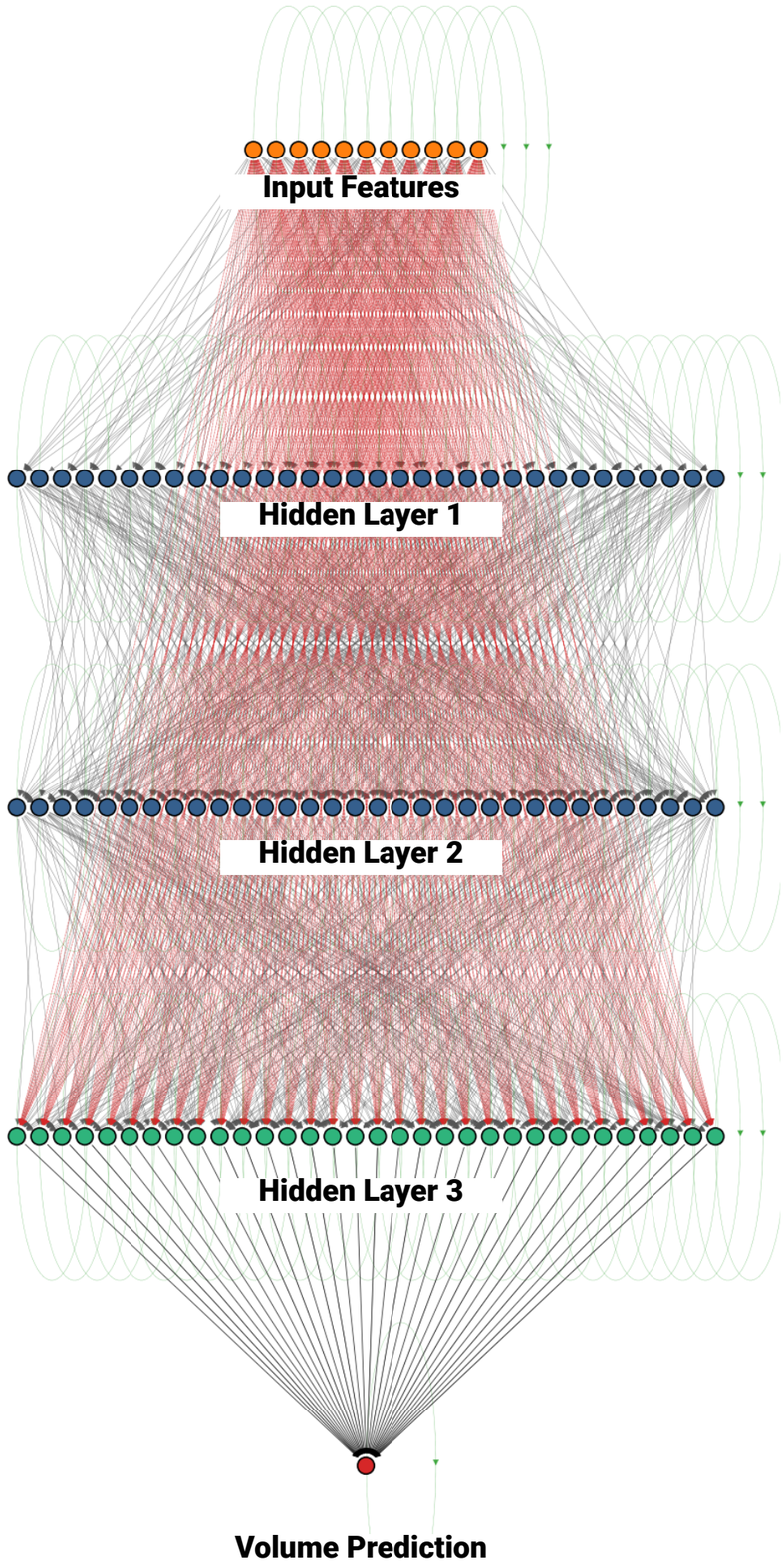


Figure 69: Adding a "skip" to the layer connections/ Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.3.1 Neural Network Architecture

To Achieve Healthier Training Results

What is dropout? Dropout is a technique in which, during each forward and backward pass, a subset of neurons is temporarily "dropped out" or set to zero. This prevents complex co-adaptations of feature detectors and encourages the network to develop multiple, redundant ways of representing important patterns. It is widely used to improve generalization in deep learning models. In more simple terms, if the model for some reason shifts the way of predictions only on some of the features eg. the building module size, then droput assists a lot for a more explorative approach since it forces the neural network to disregard some of the features randomly.

During training, this mechanism produced a noticeably smoother upward trend in validation accuracy but sudden spikes and dips occurred frequently in the beginning but over the course of learning the model steadily climbed toward its peak performance without exhibiting clear signs of under-fitting.

At the same time, the very nature of dropout means that each mini-batch presents a slightly different network architecture, so individual epochs still show small, transient drops in accuracy before recovering.

A trade-off emerged in training duration. Whereas the original network achieved its best accuracy of around eighty percent in roughly 150 epochs, the dropout-augmented model required nearly 500 to reach the mid-eighties.

In other words, total training time roughly doubled. This slower convergence reflects the added challenge the model faces in learning robust representations under the randomness of dropout, but it ultimately yields a more reliable accuracy curve and reduces the risk of over-reliance on any particular neuron.

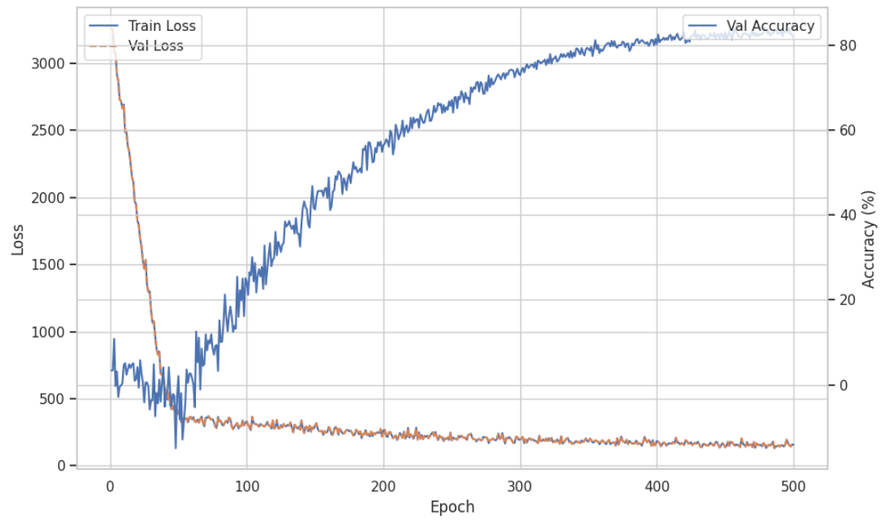


Figure 70: Increased training time but more reliable results/ Source: Author, 2025.

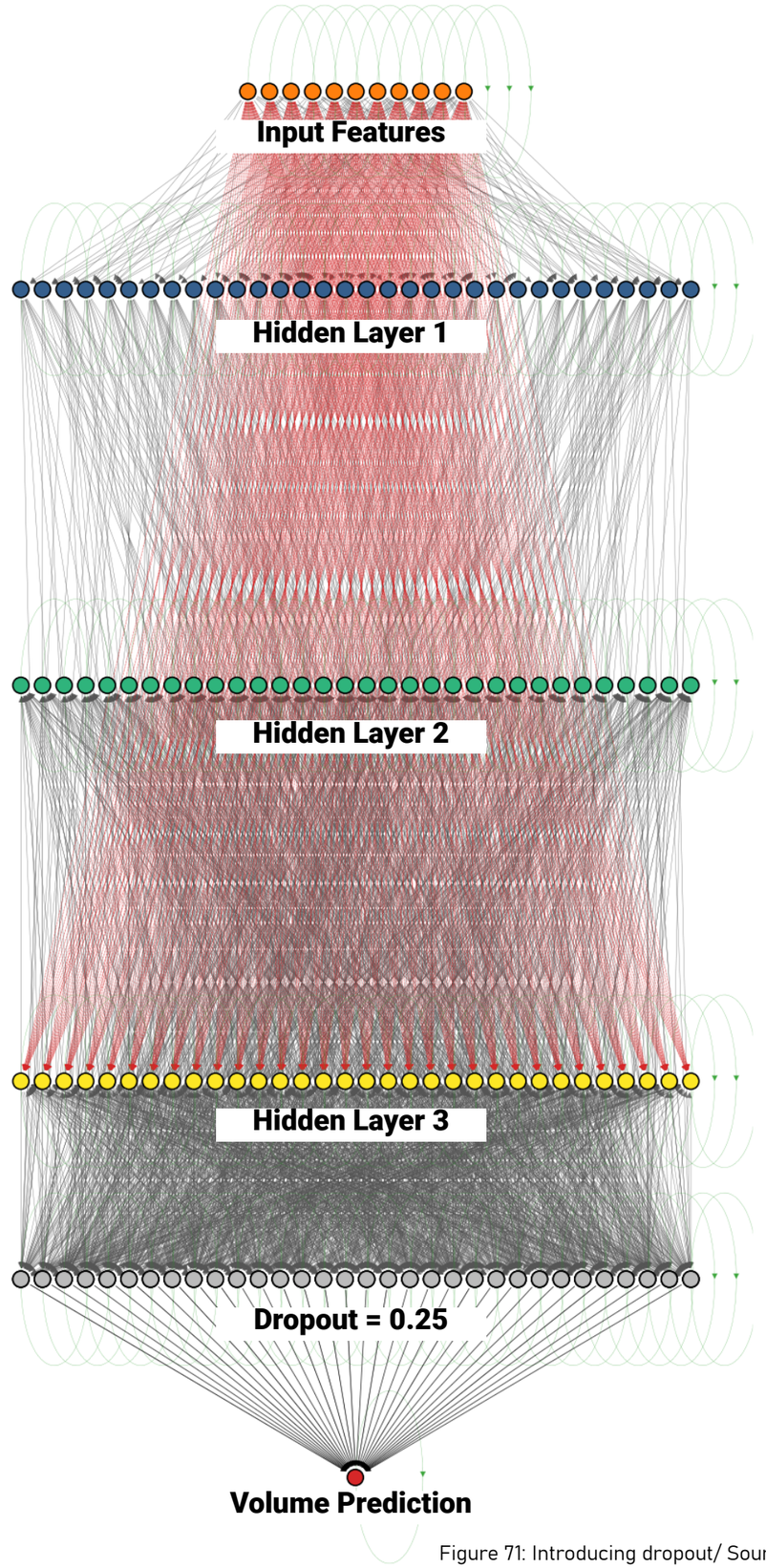


Figure 71: Introducing dropout/ Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.3.2 Loss Functions & Optimisers

A sudden, recurrent drop in validation accuracy signals that the model's learning dynamics remain unstable, even as the overall trend climbs. To push beyond the 80 percent ceiling toward 85–90 percent accuracy, it becomes essential to examine both the loss function and the optimizer, since these two components govern how the network evaluates its errors and updates its weights.

But first what is a loss function?

A loss function is a mathematical measure of discrepancy between the network's predictions and the ground-truth targets; it guides the model toward parameter configurations that minimize this error (Goodfellow, Bengio, & Courville, 2016). An optimizer then uses gradients of that loss to adjust every weight in the network, defining the pathway and speed of convergence (Goodfellow et al., 2016).

One promising change is the adoption of AdamW in place of standard Adam. Adam combines adaptive learning rates with momentum, scaling each parameter's update by its own historical gradient magnitude (Kingma & Ba, 2015). AdamW decouples weight-decay regularization from the adaptive update rule, applying true L2 regularization to the weights themselves rather than implicitly via the gradient history—a modification shown to improve generalization and reduce early-epoch volatility (Loshchilov & Hutter, 2017).

On the loss side, replacing mean absolute percentage error (MAPE) with Huber loss introduces robustness to outliers. MAPE penalizes every percentage deviation equally, so large errors can dominate gradient signals and provoke abrupt weight swings (Makridakis, 1993). **Huber loss, by contrast, treats small residuals with a squared penalty and only switches to a linear penalty beyond a threshold, damping the influence of extreme errors on the update step (Huber, 1964).**

Combined, AdamW and Huber loss yield a markedly more stable training process. Early spikes in validation accuracy are tamed, producing a smoother, steadier upward trend. As training progresses, those transient drops become progressively rarer, and overall convergence toward the mid-eighties in validation accuracy accelerates without signs of under-fitting.

Shortly:

Adam
Adapts each weight's step size individually using both momentum and RMS scaling
Mixes any weight-decay penalty into the gradient update, causing the regularization strength to vary

AdamW
Uses the same adaptive step-size and momentum rules as Adam
Applies weight decay afterward by directly shrinking each weight, keeping regularization constant

Huber Loss
Penalizes small errors with a squared term, encouraging precise fits
Switches to a linear penalty for large errors, preventing outliers from dominating updates

Figure 72: Training Performance After changing to AdamW and HuberLoss/ Source: Author, 2025.

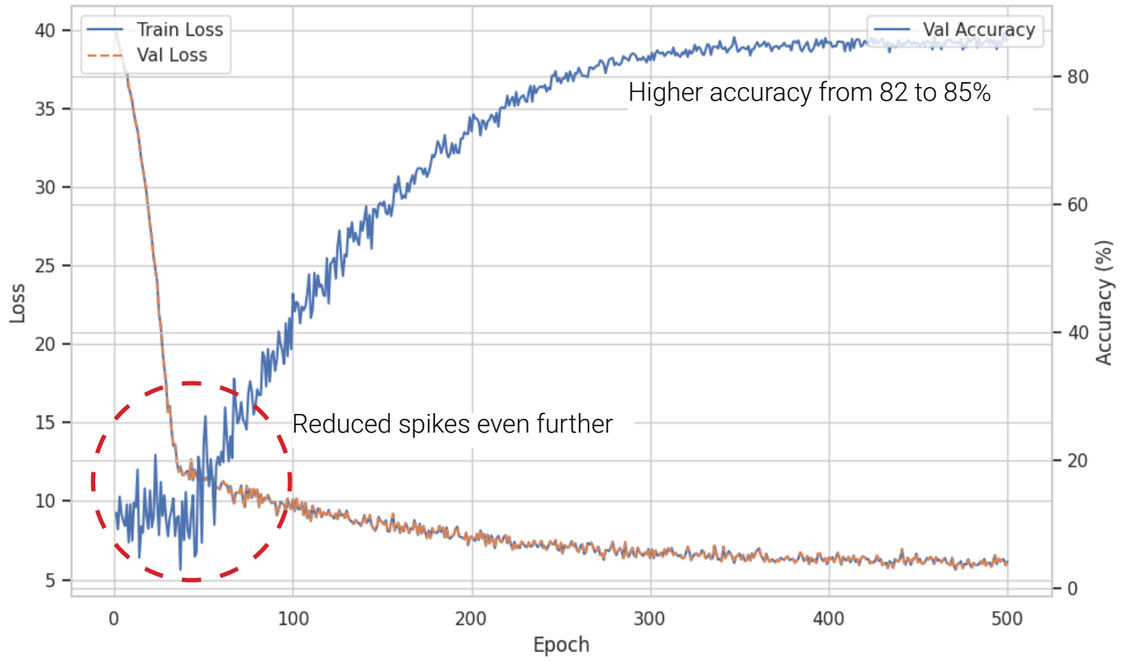


Figure 73: Hyperpameter tuning reduces the issue even further and helps achieve even higher accuracy / Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.4 Understanding The Physical Problem With Feature Contributions

This chapter examines how feature-attribution methods reveal the model's misconceptions about the physical problem.

Integrated Gradients, attribute each prediction back to the input features by accumulating the gradients along a straight path from a baseline input to the actual example. **In effect, Integrated Gradients quantify how changing a feature from zero (or any chosen reference) up to its true value contributes to the final output.**

Applying Integrated Gradients to both in-distribution and out-of-distribution designs produced heatmaps of feature contributions for every module. In these plots, warm colors indicate features that push the volume prediction upward, while cool colors indicate features that would pull it downward. When examining OOD cases, it became immediately clear that the network was assigning positive contributions to intersection area and opening area features. Physically, shared wall interfaces should reduce total material to avoid double-counting, and larger openings must subtract from volume rather than add. **The fact that the model treats these features as volume-increasing reveals a fundamental misunderstanding of the underlying geometry.**

This diagnostic insight motivates the next phase of development: reengineering either the network architecture or the feature encoding so that interaction features and opening metrics are learned with the correct sign and relative weight. Any modification must preserve the overall training stability and avoid degrading the 85-90 percent accuracy already achieved. By combining attribution-driven analysis with targeted architectural or objective adjustments, the goal is to guide the model toward a more faithful internal representation of material volume.

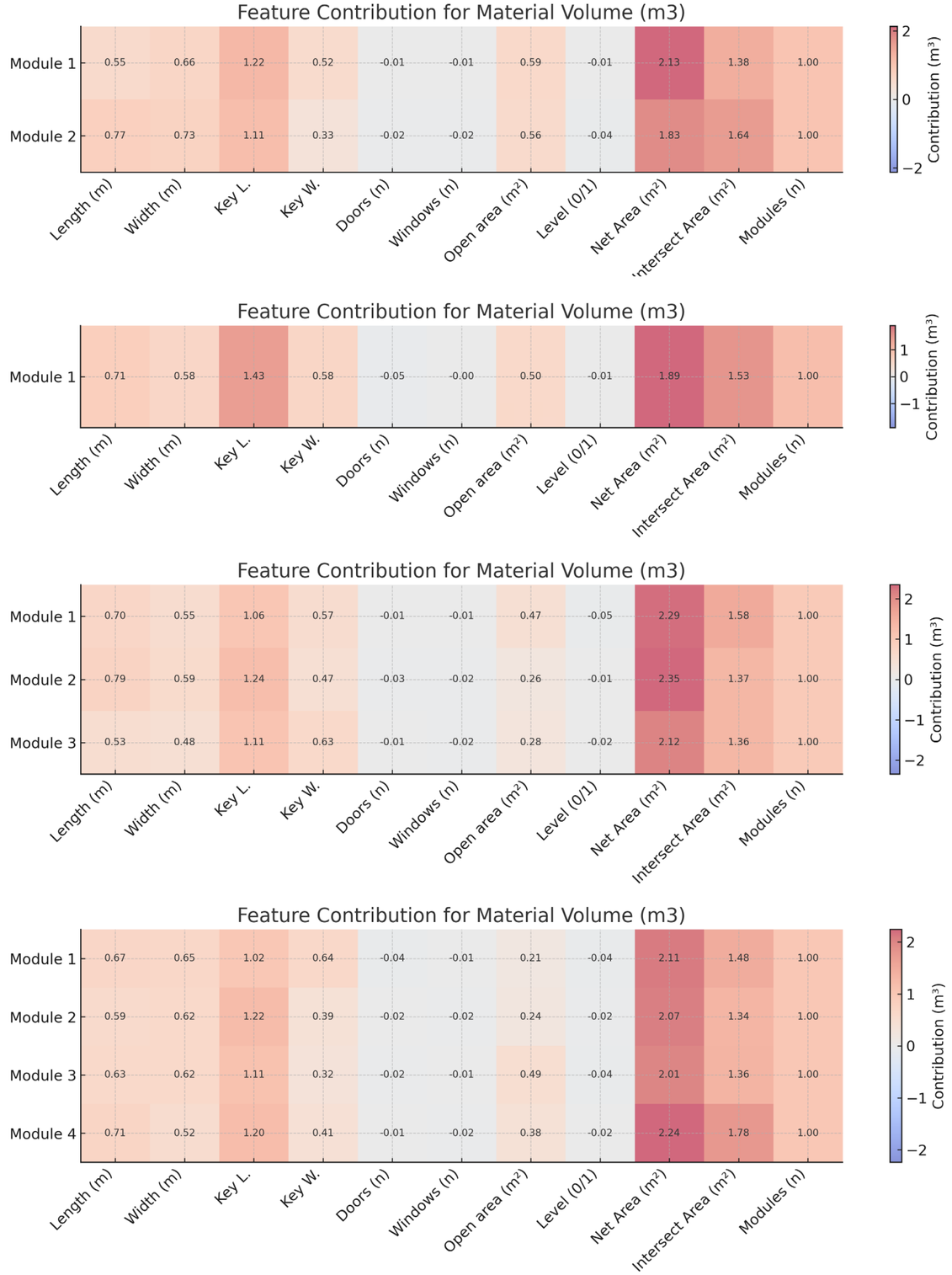


Figure 74: Visualisations of the contributions of each input feature for the volume predictions
Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.4 Understanding The Physical Problem With Feature Contributions

Developing a method of providing real-time material volume predictions which in combination with our EPD database to provide important feedback (look at chapters A,B and C), **is one of the main methods to approach the research objectives.**

But neither the most important or the only one.

Being able to connect the user-driven input with the feedback of the neural and to be able to provide the users of this computational tool with the ability to intepret how their decisions specifically affect the material usage, is one of the core components of the research.

So without affecting the already achieved high-level of accuracy the network provides, it must be reworked to also udnderstand with a realistic approach how the input features should affect the volume predictions.

Specifically:

A window, a door or an intersection between building modules where there is no material placed, technically a "void" should always be considered that these variables contribute negatively to the material volume predictions.

Which means that features like:

1)Door Count (n)

2)Window Count(n)

3)Intersection Area(n)

These 3 can never contribute positively to our output feature.

Two mechanisms guarantee that void-related features (door count, window count, intersection area) can only reduce the predicted volume. **First, a lightweight skip branch carries each node's original measurements directly into the volume output.** For the three void features, raw parameters a_i are constrained via:

$$w_i = -\operatorname{softplus}(\alpha_i) \leq 0$$

so that the model's final volume prediction becomes

$$\hat{\mathrm{H}}\mathrm{V} = V\_\mathrm{GNN} + \sum iw\_i \times \mathrm{feat}\_i$$

ensuring each w_i subtracts material rather than adds (Goodfellow, Bengio, & Courville, 2016).

Second, a sign-regularization penalty further discourages any positive influence of void features. After each forward pass, the sum of all module volumes is differentiated with respect to the input features. Any positive gradient for a void feature is penalized by

$$\mathscr{L}\mathrm{sign} = \lambda \cdot \mathrm{ReLU}\left(\frac{\partial \sum \hat{\mathrm{H}}\mathrm{V}_n}{\partial\, \mathrm{feat\ void}}\right)$$

which enforces a negative contribution for every void channel (Goodfellow et al., 2016).

So what is the proposed GNN structure?

1.Pre-Embedding MLP
A small "multi-layer perceptron" (MLP) applies two simple fully connected layers with a nonlinear activation in between. In plain terms, this is a tiny neural network that learns how to combine the eleven raw input features (length, width, door count, etc.) into a more useful 32-dimensional summary before any graph processing begins (Goodfellow, Bengio, & Courville, 2016).

2. Graph Attention Layers (GAT)
Three successive "graph attention" modules perform the core message-passing. Each module (node) gathers information from its neighbors, but rather than treating all neighbors the same, the attention mechanism learns which neighbors are most relevant to volume prediction. Concretely, every connection is assigned a learned weight, so that adjacent modules with stronger structural influence contribute more to the node's updated representation (Veličković et al., 2018).

3. Node-Wise Normalization
After each attention step, the node features are normalized to zero mean and unit variance. This per-node normalization prevents any single feature vector from dominating, keeping the scale of activations consistent across diverse graph structures (Ioffe & Szegedy, 2015).

4. Residual and Dropout Connections
a)Residual (skip) connections add each layer's output back into its input for the next layer, ensuring original information is never lost and enabling deeper architectures to train efficiently (He et al., 2016).
b)Dropout randomly silences 35 % of the hidden units during training, encouraging the network to develop multiple redundant pathways and smoothing out sudden accuracy spikes (Srivastava et al., 2014).

5.Monotonic Decreasing Skip For Void Features
In parallel with the graph layers, a direct linear branch carries the pre-embedded features forward. Three learned weights,one each for door count, window count, and intersection area are constrained to be non-positive, so that when these void measurements reach the final volume output they always subtract from the prediction, enforcing the correct physical behavior.

6. Output Layer
A final linear layer takes the combined graph and skip representations and produces 1 number per module: **the predicted LVL volume.**

E.A Graph Neural Network For Predicting Material Volume

E.4 Understanding The Physical Problem With Feature Contributions

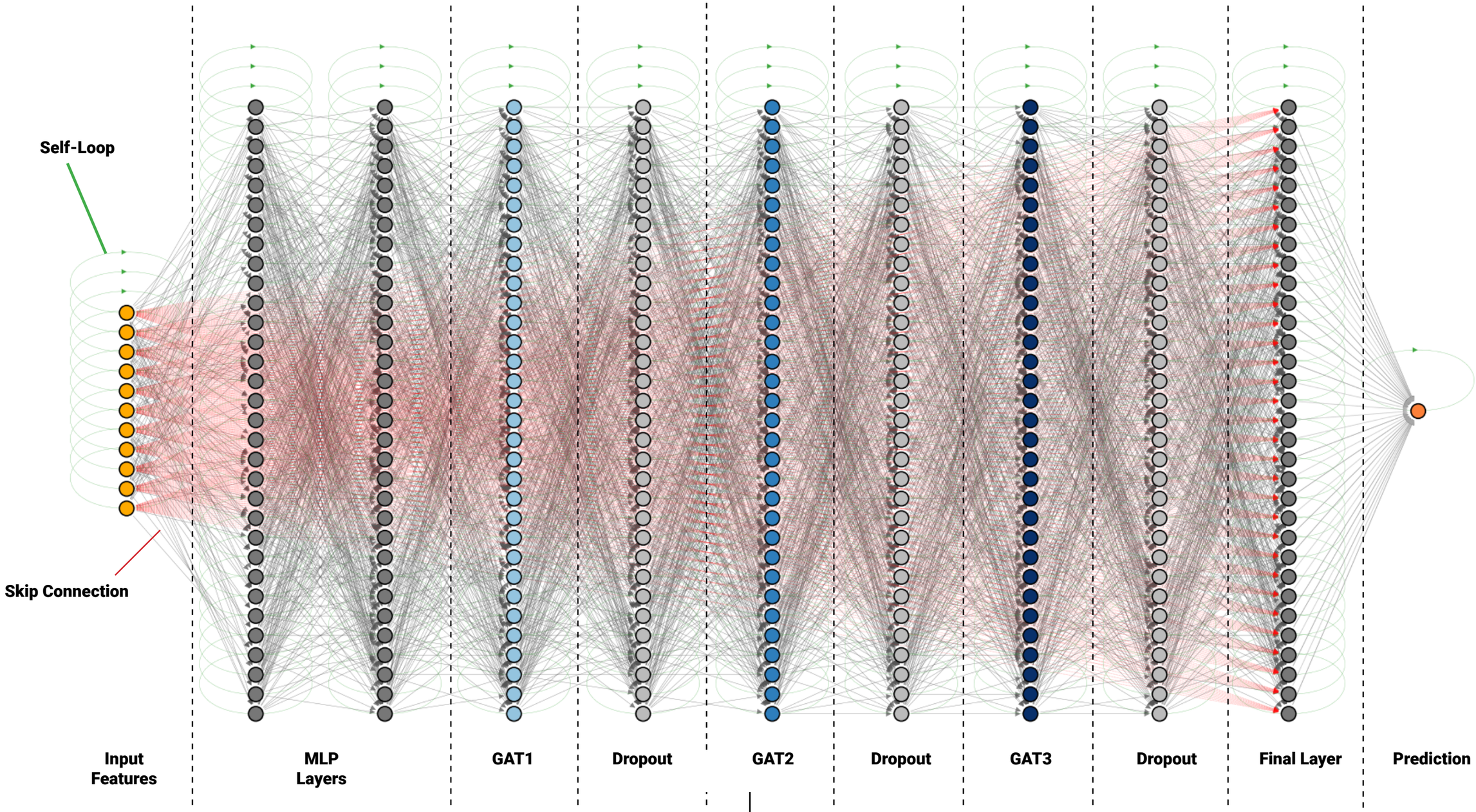


Figure 75: The proposed GNN structure
Source: Author, 2025.

GRAPH NEURAL NETWORK DEVELOPMENT

E.A Graph Neural Network For Predicting Material Volume

E.4 Understanding The Physical Problem With Feature Contributions

After integrating the monotonic-decreasing skip branch and sign-regularization penalty, the training dynamics change markedly (Figure 82). No longer does validation accuracy suffer abrupt drops; instead, the accuracy curve climbs smoothly toward the mid-eighties percent without the sudden up-and-down spikes seen previously.

At the same time, convergence slows, rather than reaching its peak within a few dozen epochs, the network continues to refine its weights over the full 300-epoch schedule. This slower march to optimality reflects the fact that, whenever the model attempts to assign a positive contribution to any void feature, the sign penalty pushes it back below zero. In effect, the network must explore a broader range of internal pathways, seeking alternate combinations of attention weights and skip connections that satisfy both the volume-prediction objective and the enforced subtraction semantics.

The cost of this expanded search is a modest increase in training time, but with two clear benefits. **First, the void features (door count, window count, intersection area) now consistently carry negative attribution in the scaled-contribution heatmaps (Figure 83), aligning the model's internal logic with physical reality.**

Second, the absence of erratic accuracy dips builds confidence that the GNN will behave predictably even when presented with novel module assemblies. In sum, the penalized monotonic network trades off a bit of speed for a robust, interpretable mapping from architectural features to material-volume estimates.

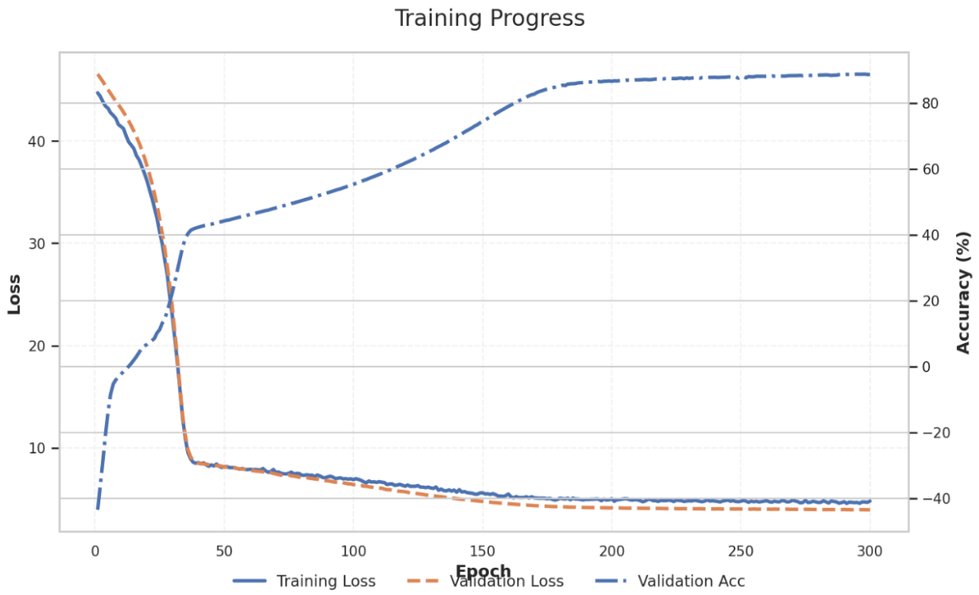
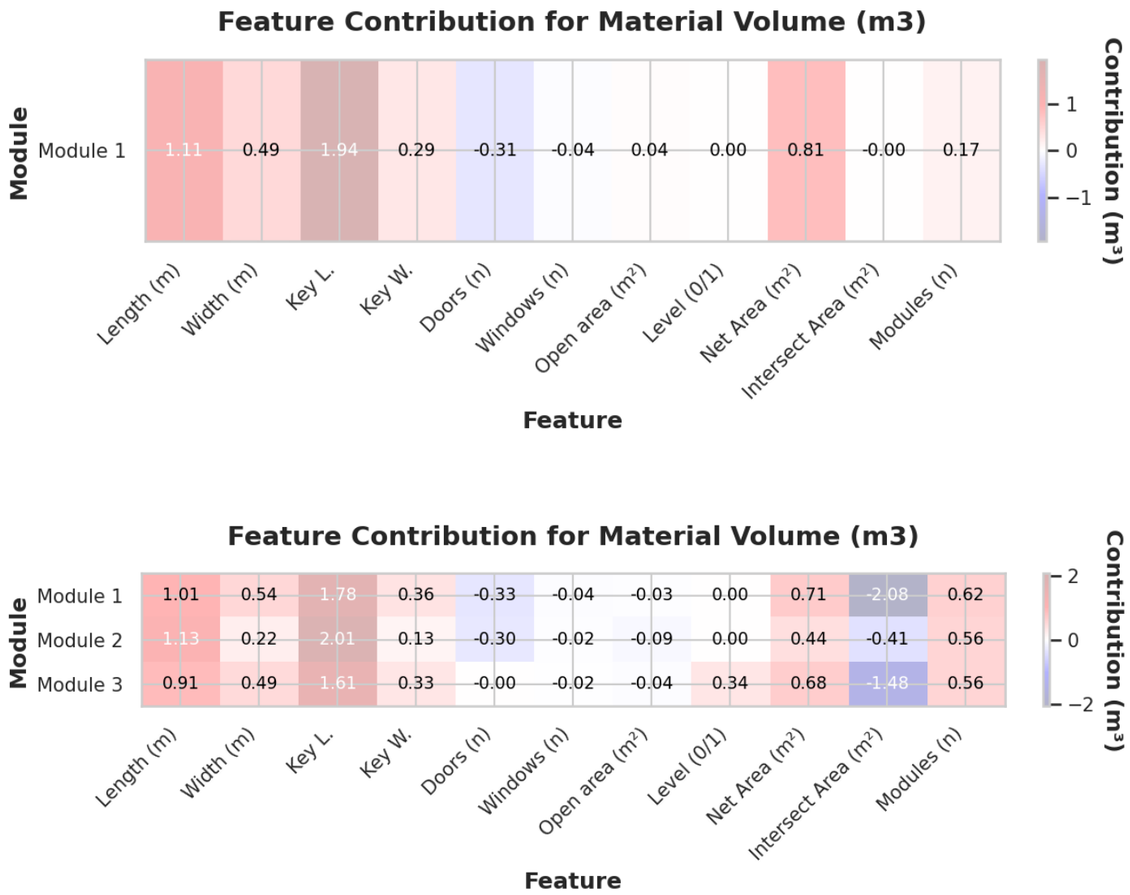


Figure 76: No rapid convergence, still reaching 85-87% accuracy, Source: Author, 2025.



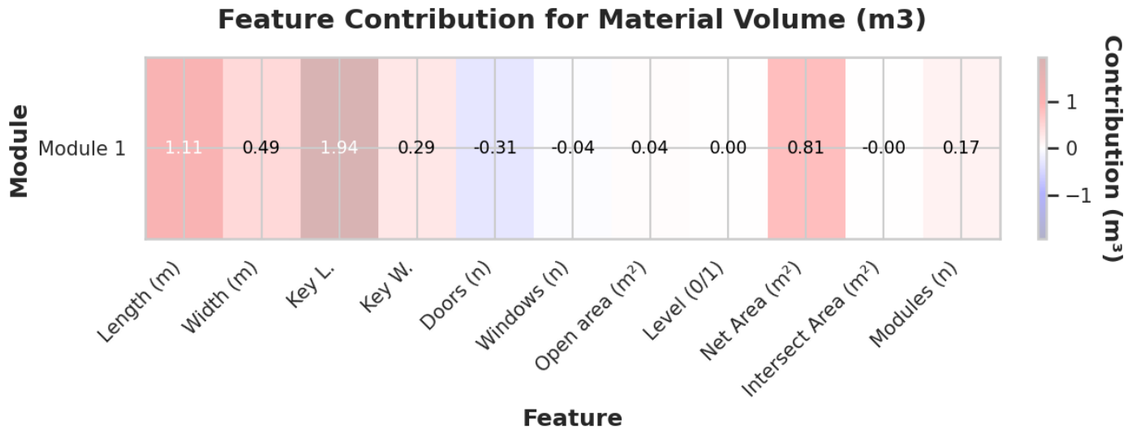


Figure 77: The feature contributions showing a more realistic result, Source: Author, 2025.

E.A Graph Neural Network For Predicting Material Volume

E.5 Conclusions

The work presented in this chapter directly addresses the core research problem of embedding explainable, real-time life-cycle assessment into early-stage architectural design. By iteratively refining the Graph Neural Network, the following outcomes were achieved:

1.Accurate, Stable Prediction (Objective 2)
Initial under-fitting and erratic accuracy on out-of-distribution designs revealed that a basic three-layer GNN lacked the capacity and inductive biases to capture both local and global volume relationships. Introducing dropout, AdamW optimization, and Huber loss stabilized training and raised in-distribution accuracy into the mid-eighties percent. The final monotonic-decreasing skip branch and sign-regularization penalty preserved per-module uniqueness and enforced physically correct subtraction semantics for void features, all without sacrificing predictive performance.

2.Transparent Feature Attribution (Objective 3)
Integrated Gradients exposed the model's early misconceptions—treating openings and intersections as volume-increasing—underscoring the necessity of embedding explanation mechanisms. The monotonic network now produces negative attributions for void features, aligning internal logic with real-world physics and providing designers clear, component-level insights into how each parameter drives the predicted volume.

3.Integration with LCA Data (Objective 1)
By learning to predict module-wise LVL volumes from minimal geometric and connectivity inputs, the network becomes a lightweight surrogate for full Sustainer geometries. Coupled with an existing EPD database, this enables instantaneous embodied-carbon calculations that reflect project-specific configurations, closing the gap between concept sketches and rigorous life-cycle metrics.

4.Real-Time, Explainable Feedback (Objective 4)
The stabilized GNN, equipped with feature-attribution visualizations, can be embedded within a design interface to deliver immediate, interpretable feedback. Every adjustment to box dimensions, opening placements, or module adjacency triggers both updated volume predictions and explanatory heatmaps, empowering architects to make informed, sustainability-driven decisions on the fly.

5.Scalability and Generalizability (Objective 5)
The final model demonstrates robust behavior on both in-distribution and out-of-distribution assemblies, validating its applicability across diverse modular configurations. The use of standard graph layers, combined with targeted architectural constraints, ensures that the approach extends naturally to larger or more complex systems without retraining from scratch.

In summary, the enhanced GNN framework transforms circular-economy aspirations into an actionable design-stage tool. By integrating high-fidelity LCA data, delivering accurate and stable volume predictions, and exposing clear explanatory pathways, this work paves the way for architects to embed genuine life-cycle thinking into the earliest phases of ideation—shifting sustainability from a late-stage compliance check to a proactive, design-driving principle.

End Of Chapter E. A Graph Neural Network For Predicting Material Volume

Work Package 04 | Tool Development & Integration

F. Echo | An Intelligent Design Assistant

Brief Summary

Chapter F describes the development and integration of Echo, an intelligent design assistant built as a suite of Grasshopper components inside Rhino 8. Beginning with a Project Setup node that binds user-specified site, manufacturer, LCA, and end-of-life parameters to the EPD database and transport-distance calculations, the pipeline proceeds to a Material Calculator that extracts geometric features, builds in-memory graphs, and runs the pre-trained GNN to produce per-module volume predictions and associated life-cycle metrics. A predict-only variant streamlines inference by loading frozen model weights and bypassing any training logic; visualizations—statistics tables, holistic LCA charts, feature-attribution heatmaps, and carbon-performance comparators—refresh instantly as designers tweak their conceptual layouts. By packaging data retrieval, neural inference, and interpretable feedback into modular, environment-agnostic nodes, Echo delivers on the thesis objectives of embedding live LCA metrics, enabling real-time GNN predictions, and surfacing transparent explanations directly within the architect's familiar visual scripting environment.

F. Echo | An Intelligent Design Assistant
F.1 Shifting Development Environments

Integrating the trained GNN and EPD database into an architect's workflow required embedding the predictive engine directly within the Rhino / Grasshopper environment. Beginning with Rhino 8, the platform's migration from .NET Framework to .NET Core and adoption of Python 3 for Grasshopper scripting enabled the development of self-contained Python components that execute seamlessly inside the visual programming canvas. **This shift eliminated the need for external executables or inter-process communication (like external 3rd party plugins), simplifying deployment.**

However, several challenges arose.

Rhino's native geometry types (NURBS curves, Breps, points) differ from the lightweight graph abstractions used by the GNN. To bridge this gap, a minimal preprocessing step was implemented: each building module is represented as a simple 3D box defined by its corner points, and openings are converted to planar curves. These primitives serve as the sole inputs to the Grasshopper Python component, which then constructs the required feature vectors and connectivity lists for the GNN.

To maximize interoperability and future scalability, the component's public interface exposes only basic geometric inputs—boxes and curves—so that Echo variants can be developed for other CAD or BIM platforms.

All Rhino-specific API calls are confined to a thin wrapper layer that translates native geometry into the standardized JSON schema consumed by the GNN. Prediction results and feature-attribution heatmaps are then rendered back into Grasshopper as data trees and preview geometry, leveraging standard visualization methods rather than proprietary UI elements.

This environment-agnostic design ensures that, although the current implementation runs inside Rhino/Grasshopper, the core predictive functionality and data-flow logic remain portable. As a result, future extensions can target alternative modeling tools or web-based interfaces without reengineering the underlying GNN pipeline.

One of the first challenges was enabling the GNN to execute in a "predict-only" mode inside Grasshopper, delivering near-instant feedback on material volume without retraining. In this variant, the component performs two tasks.

1. Model Loading and Inference
A now already trained neural network (WP3) exported once as a .pth file is loaded at runtime. The Grasshopper Python component skips all training routines, instantiates the network architecture, and restores its weights from the specified file path. When the user supplies a set of numerical input features (module dimensions, opening counts, etc.), the component reconstructs the graph in memory and calls the model's forward method to compute volume predictions in milliseconds.

2.Standalone Prediction Workflow
The original script assumed JSON files would already exist on disk. For a pure inference tool, Grasshopper must consume raw geometry rather than pre-generated JSON. Therefore, the predict-only component expects:

-A user-designated folder containing the .pth model file

-A Grasshopper data tree of 3D boxes (modules) and planar curves (openings) When triggered, the script reads the boxes and curves, extracts the same eleven input features used during training, and internally builds the JSON-like dictionaries. These dictionaries are immediately converted into DGL graphs, after which the model returns volume estimates. No external JSON files are written or read—everything occurs in memory.

But to merge the functionality of Work Package 02 (training-design generation) with Work Package 03 (GNN development), a single Grasshopper component was developed that performs both feature extraction and inference:

1.Geometry Feature Extraction
The component iterates over each provided 3D box and its associated curves. Length, width, height, opening counts, total opening area, intersection areas, level, and module count features are computed on the fly and assembled into an in-memory JSON structure identical to the training schema.

2.JSON Graph Construction
Without writing to disk, the JSON structure is passed to the same routine used in training. This guarantees that node indexing, edge lists, and self-loops match the network's expectations.

3.Volume Prediction
With the graph assembled, the pre-loaded GNN performs forward inference, returning volume predictions. Results are output as Grasshopper generated charts.

By unifying these steps, the pipeline allows a user to sketch modules and openings in Rhino/ Grasshopper, click a single button, and receive both numerical volume feedback and visual attributions leveraging all prior research outputs in a cohesive, real-time tool.

F. Echo | An Intelligent Design Assistant
F.2 The Grasshopper Components

So what are the components need to be developed to align with the research's objectives?

The Grasshopper implementation of Echo is organized as a sequence of interoperable components that together realize the unified pipeline described in F.1. Each component corresponds to a distinct stage in the workflow, from project configuration to feature extraction, graph construction, GNN inference, and visualization ensuring that designers can move seamlessly from a simple geometric sketch to actionable life-cycle feedback.

By encapsulating each responsibility in a dedicated node, the system remains modular, extensible, and agnostic to the upstream authoring environment, while still delivering the full suite of research outputs in real time.

1. Project Setup Component
The Project Setup component serves as the primary filter for all downstream visual feedback (aside from volume predictions). It accepts four user parameters:

-Project Address :string for Open Street Map
-Manufacturer ID : integer which corresponds to the proper sheet in the EPD database
-LCA Scenario: integer which filters between the 3 lca scenarios and which data cells to pull information from the EPD database
-End-of-Life Scenario: same as LCA scenario

Upon receiving these inputs, the component performs two functions. First, it queries the integrated EPD database to retrieve the matching records for the selected manufacturer, LCA, and EOL scenarios, exposing these outputs:

-EPD_ID, LCA_ID: selected integers (used as input in other components)
-EPD_info: Informs the user on what who is the selected manufactuers with some additional benchmark info.
-LCA_info: Informs the user on the selected LCA scenario (cradle to gate, cradle to grave or cradle to grave and beyond)
-EOL_info: Same as LCA_info (reuse, recycle,landfill, incineration), although if the user chose cradle to gate as the selected lca scenario then an EOL scenario is not applicable since the study ends manufacturing.

Second, it computes the transportation distance from the project address to the manufacturer's location, using the methodology detailed in Chapter C and outputs a project distance value in km. These outputs both configure subsequent components (ensuring that material and carbon calculations use the correct data) and provide designers with immediate contextual information about their choices, grounding the volume-prediction pipeline in real-world logistics and environmental parameters.

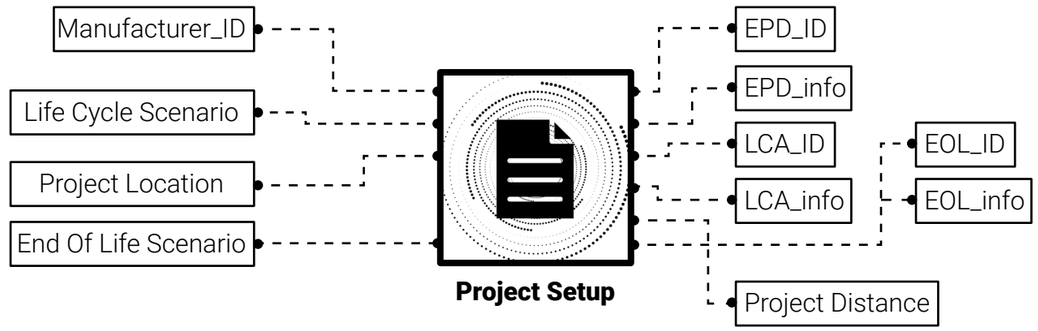


Figure 78: The project setup component, Source: Author, 2025.

ECHO | AN INTELLIGENT DESIGN ASSISTANT

F. Echo | An Intelligent Design Assistant
F.2 The Grasshopper Components

2. Material Calculator Component
The Material Calculator is the core computational engine of Echo, responsible for converting inputs from the Project Setup and the user's geometric definitions into the full suite of material and environmental outputs. It accepts:

-EPD_ID, LCA_ID, EOL_ID (from Project Setup)
-Project Distance (from Project Setup)
-Modules (3D boxes)
-Openings (planar curves)
-Toggle A / Toggle B (booleans)

Upon execution, this component performs the following sequence (analyzed in previous section):
1.Feature Extraction & Graph Construction
2.Volume Prediction via GNN

a)Material & Carbon Calculations
-Total LVL Volume is summed across all modules.
-Volume Per Module is preserved for granular feedback.
-Embodied Carbon is computed by multiplying module volumes by the cradle-to-grave from the selected EPD data.
b)Sequestrated Carbon:reflects the biogenic carbon stored in the LVL panels.
c)Renewable : Non-Renewable Source Ratio: is derived from the EPD's energy consumption.
d)Transportation Emissions:are calculated using Project Distance and the EPD's logistics factors.
e)Feature Contributions & Toggles: When Toggle A is enabled, the component invokes the trained GNN and it start working in real time to produce predictions. Toggle B is specifically for the feature contributions, because it is a slightly more computational heavy process (takes about 10-15 seconds, meaning it's not instant).
f)Stored Values:Instead of a direct calculation output, this output comes in the form of a data tree which as a collective storage for all of the calculations. This later works as input for the visualisation components.

By encapsulating model inference, life-cycle computations, and interpretability tools in a single Grasshopper node, the Material Calculator delivers all necessary outputs—numerical and visual—required for informed, sustainability-driven design decisions directly within the architect's familiar visual scripting environment.

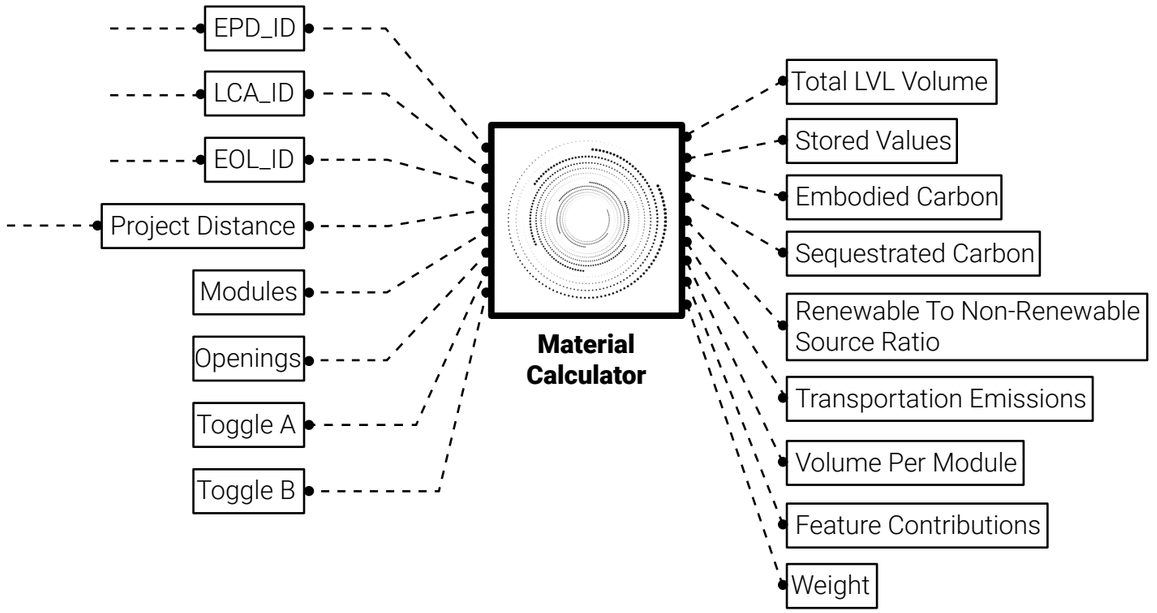


Figure 79: The main components that computes most of the necessary outputs, Source: Author, 2025.

F. Echo | An Intelligent Design Assistant
F.3 Visualisation Feedback

To complete the Echo pipeline, a suite of visualization components presents predictive and environmental data in an immediately digestible format. All visualizations update in real time as the user adjusts module geometry, opening placements, or project parameters, reflecting changes in the selected LCA and EOL scenarios.

1. Statistics Panel

Displays a per-module breakdown of key metrics:

-Dimensions (length × width × height)
-Predicted LVL Volume (m^3)
-Embodied Carbon (kg CO_2 eq)
-Sequestered Carbon(kg CO_2 eq)
-Transportation Emissions (kg CO_2 eq)
-Renewable : Non-Renewable Source Ratio
-Module Weight(kg)

Whenever the conceptual geometry or project settings change, the panel refreshes to show updated values under the current LCA and EOL scenarios.

2.Holistic LCA Chart

A line chart illustrating the full cradle-to-grave carbon profile (phases A1–D) for the entire design. Although the EPD and chosen scenarios influence unit factors, this view always decomposes total emissions into each lifecycle stage. Modifications to the design geometry or end-of-life selection immediately shift the height and color distribution of the chart.

3. Feature Contributions Heatmap

Visualizes how each of the eleven GNN input features contributes to the predicted volume for every module (in m^3). Positive (warm) and negative (cool) contributions are color-coded, allowing the user to diagnose which parameters—such as opening area or intersection contact—drive volume estimates. The heatmap recalculates after every design tweak.

4. Carbon Performance Comparator

Offers two modes:

-Module Comparison: Side-by-side bar charts comparing embodied carbon across selected modules.
-Design Aggregate: Single-value display of total carbon emissions for the entire assembly.

Designers can toggle between modes to focus on hotspot modules or assess overall performance at a glance. Changes to geometry, material choices, or logistics instantly propagate through this component.

Together, these visualization tools transform raw GNN outputs and life-cycle data into actionable insights, empowering architects to explore trade-offs and optimize circular-economy objectives from the earliest stages of design.

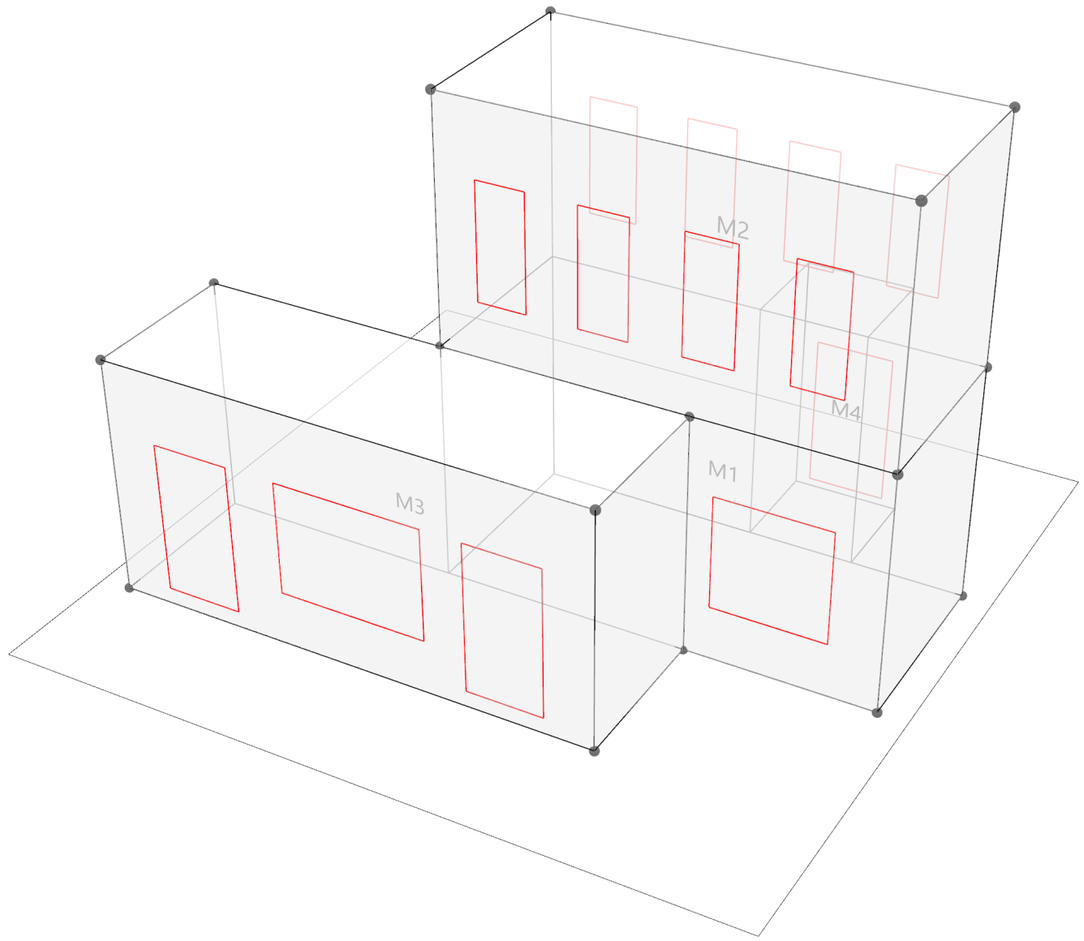


Figure 80: Once the data is inputted the material calculator loops through the design to name the building modules, Source: Author, 2025.

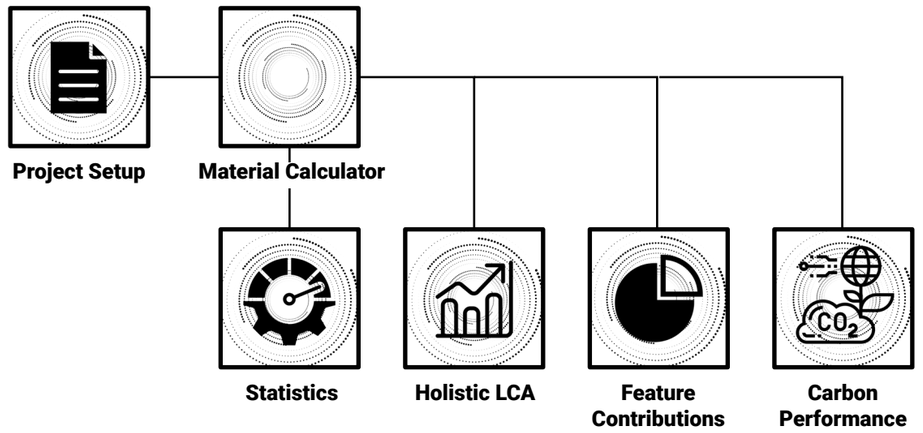


Figure 81: Grasshopper components overview, Source: Author, 2025.

F. Echo | An Intelligent Design Assistant

F.3 Visualisation Feedback

Here, you can see a Rhinoceros 3D viewport where most of the grasshopper components are active to provide real-time feedback.

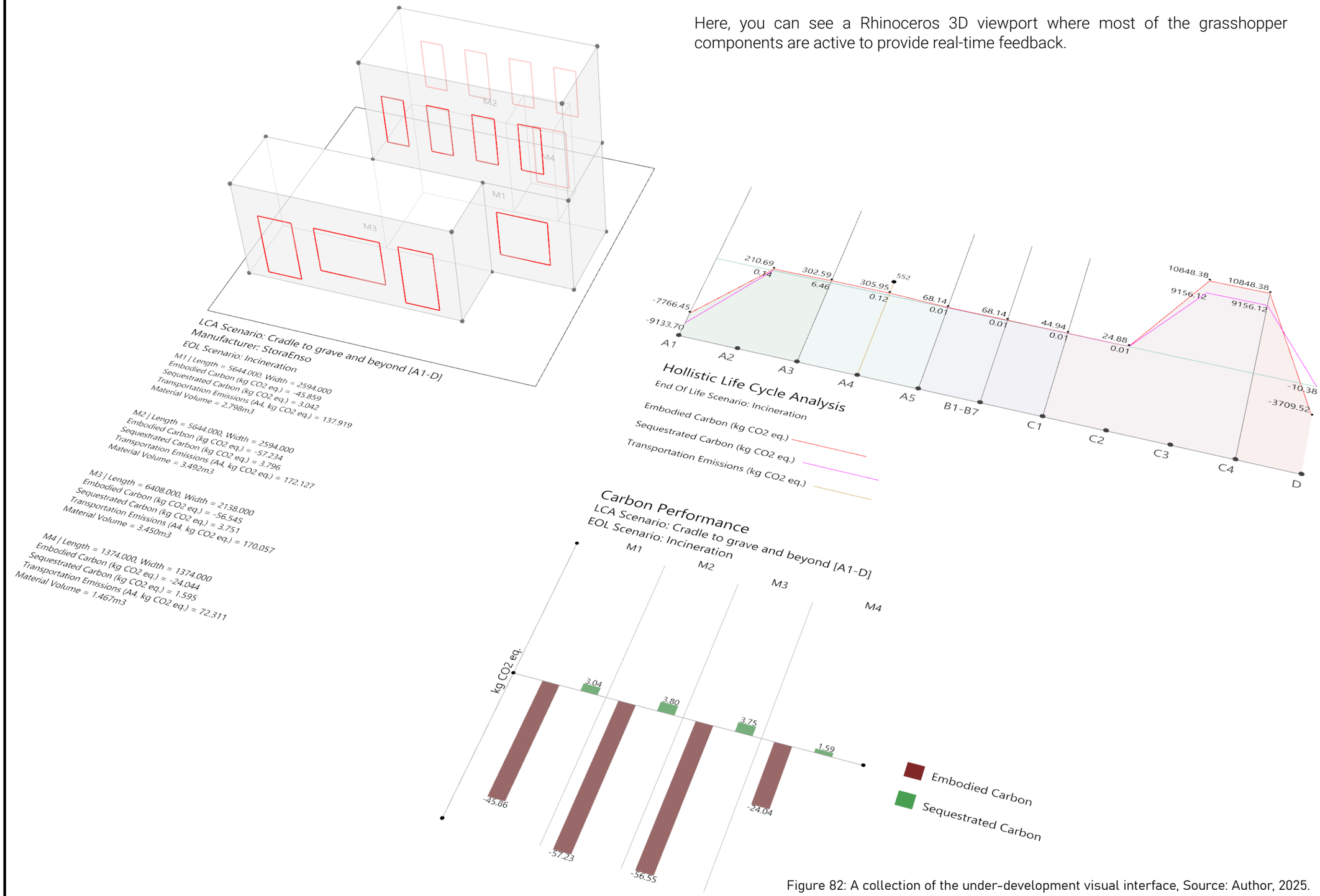


Figure 82: A collection of the under-development visual interface, Source: Author, 2025.

ECHO | AN INTELLIGENT DESIGN ASSISTANT

F. Echo | An Intelligent Design Assistant
F.3 Visualisation Feedback

User Interface Generated From Components Preview (Updated)

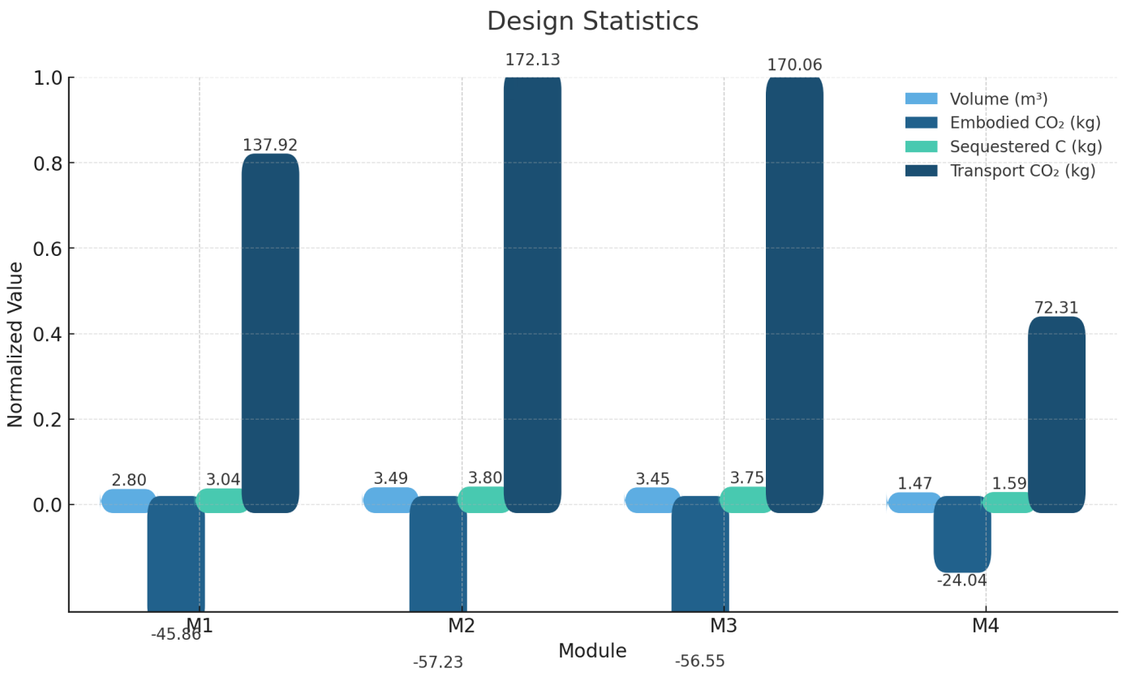


Figure 83: Updated Design Statistics Interface, Source: Author, 2025.

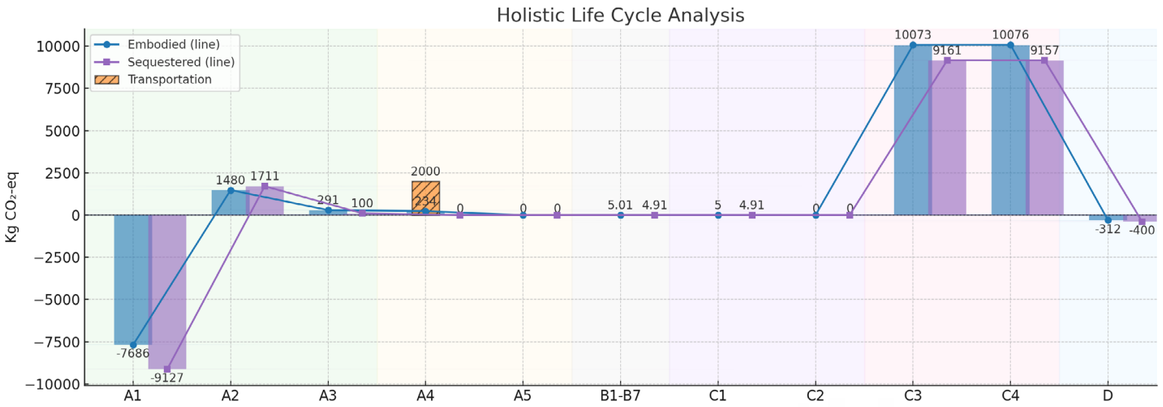


Figure 84: Updated Holistic LCA Interface, Source: Author, 2025.

F. Echo | An Intelligent Design Assistant
F.4 Conclusions

The Grasshopper–embedded Echo pipeline directly addresses each of the thesis's core objectives by delivering a seamless, real-time, and explainable life-cycle assessment tool within the conceptual design environment.

Objective 1: Integrated LCA Framework
The Project Setup and Material Calculator components realize a unified LCA framework (Objective 1) by binding the EPD database to geometry inputs and automating the computation of embodied carbon, sequestered carbon, renewable/non-renewable energy ratios, and transportation emissions. Designers no longer must manually sift through spreadsheets or disparate databases—every life-cycle metric is calculated on demand from the selected manufacturer, LCA, and EOL scenarios.

Objective 2: GNN Model Development and Calibration
Embedding the pre-trained GNN within Grasshopper fulfills Objective 2, enabling sub-second volume predictions for arbitrary module assemblies. By reconstructing the node-edge graph in memory and loading frozen model weights, the Material Calculator node demonstrates that the calibrated network achieves the target 85–90 % accuracy in situ, without retraining, directly on user-supplied geometry.

Objective 3: Explanation Mechanisms
Toggle-driven feature-attribution heatmaps, generated on demand via Captum's Integrated Gradients, satisfy Objective 3 by surfacing per-feature contributions to each module's volume. Warm and cool color codes immediately reveal when design choices—such as openings or interface areas—introduce physically inconsistent predictions, guiding users toward corrective refinements.

Objective 4: Real-Time Interface Integration
All visualizations,the Statistics panel, Holistic LCA chart, Feature Contributions heatmap, and Carbon Performance comparator, update instantaneously as module geometry, openings, or project parameters change. This direct integration (Objective 4) transforms circular-economy insights from a post-hoc audit into a live design driver.

Objective 5: Scalability and Generalizability
Finally, by confining all Rhino-specific logic to a thin wrapper and exposing only generic box and curve inputs, Echo's architecture remains portable and extensible (Objective 5). Future ports to alternative CAD, BIM, or web-based platforms can leverage the same Python GNN and EPD-driven workflow without reengineering the core predictive and data-flow logic.

In sum, the Grasshopper implementation of Echo achieves the thesis's ambition: it embeds a calibrated GNN and EPD-powered LCA into early-stage design, provides transparent, per-module insights, delivers feedback in real time, and does so in a modular, scalable manner that can evolve alongside emerging tools and materials.

Work Package 1

Work Package 1

Work Package 2

Work Package 2

Work Package 3

Work Package 4

WORK PACKAGE 5

Work Package 5
Research Outcomes & Contributions

G. Research Output & Reflection
G.1 Reflection

Project Overview and Methodological Reflection:
This thesis developed Echo, a computational design support tool that integrates a Graph Neural Network (GNN) with an Environmental Product Declaration (EPD) database to provide real-time, explainable life-cycle assessment (LCA) feedback for modular timber building designs.

The chosen methodology combined data-driven machine learning and building performance analysis:

A custom database of EPD data (covering four manufacturers' Laminated Veneer Lumber products) was created, a training dataset of hypothetical modular assemblies was generated via Latin Hypercube Sampling, and a GNN model was trained and calibrated to predict material volumes and environmental impacts from graph-structured building data. These components were implemented in a Grasshopper plugin, enabling designers to input conceptual geometry and immediately see predicted LCA metrics (e.g. embodied carbon) along with visual feedback highlighting the contributions of different elements. **Overall, the selected methods proved largely effective in achieving the project goals. The GNN-based approach successfully produced rapid predictions that approximate full LCA results, confirming that a graph learning model can capture the relationships in a building assembly (e.g. how module configurations influence material quantities and impacts).**

Key to this success was the refinement of the model through techniques like dropout regularization (improving generalization), monotonic skip connections and sign-penalty regularization (ensuring the model's behavior aligns with physical expectations, such as impacts increasing with added material). **These measures were introduced in response to intermediate findings and feedback, and they improved the model's reliability.** The use of Latin Hypercube Sampling to create a broad, balanced training set was justified and effective: LHS efficiently sampled the high-dimensional design space to provide diverse examples for training, which helped the GNN learn generalizable patterns.

The decision to integrate the tool into Grasshopper also worked well, as it placed the LCA feedback directly into an early-design environment familiar to architects, facilitating an intuitive design workflow. **This choice was made to ensure the research outcomes would be immediately accessible within design practice, and indeed it allowed continuous testing of the tool in a realistic setting throughout development**. Some challenges were encountered and mitigated: for example, early versions of the GNN sometimes gave implausible results (due to the black-box nature of standard neural networks), so explainability features were added to increase transparency. By breaking down impacts by building module and adding interface elements that explain which inputs drive the results, the tool became more trustworthy and useful to designers.

Feedback from supervisors throughout the process was incorporated to refine both the technical approach and the novelty of the topic itself ,for instance, comments on initial model accuracy led to additional training data generation and hyperparameter tuning. Through this iterative process, the importance of combining engineering knowledge with machine learning was highlighted: purely data-driven models benefit greatly from domain-based constraints (such as monotonicity for material impacts) to ensure realistic outcomes. **It was also learned that early-stage design tools must balance detail with speed,the approach had to be kept lean (using a streamlined GNN) to provide real-time feedback**. This trade-off was continually considered, and the outcome is a tool that provides designers with quick, reasonably accurate insights, serving as a prompt for more sustainable thinking without replacing detailed analysis later.

Relationship to the Building Technology Track and MSc Programme:
The project's topic and approach are well-aligned with the Building Technology (BT) track and the broader MSc Architecture, Urbanism and Building Sciences programme. At its core, the BT track emphasizes the integration of advanced technologies and scientific methods into building design and construction.

Echo addresses this by combining state-of-the-art computational techniques (a GNN-based predictive model) with practical building performance evaluation (life-cycle assessment of materials). In doing so, it exemplifies how digital innovation can inform sustainable building practices, a key concern in contemporary building technology research.

The focus on modular timber construction is particularly relevant to BT: timber engineering and prefabrication are cutting-edge topics in sustainable construction technology, and the project contributes by providing a tool to optimize such systems environmentally. Moreover, the MSc programme encourages a synergy of design, sustainability, and engineering—this project sits at that intersection.

It leverages architectural computing (parametric modeling in Grasshopper) to serve environmental performance goals, reflecting the programme's ethos of using building science to enhance design outcomes. By developing a design-phase LCA tool, the work shows how architects and engineers can collaborate through shared tools and data, resonating with the multidisciplinary nature of Architecture, Urbanism and Building Sciences.

In essence, the graduation project extends the knowledge domain of the BT track (which often deals with new materials, structural systems, and environmental technologies) by introducing a novel method for real-time environmental feedback. It reinforces the programme's objective of educating architects who are technically adept and environmentally conscious.

The project also responds to the educational goal of the BT track by demonstrating an innovative applied research: it not only explores a theoretical framework (AI for LCA) but also delivers a tangible solution in a design context, thereby contributing to both academic inquiry and practical architectural methodology.

Reciprocal Influence of Research and Design:
Throughout this project, research and design have continually informed and shaped each other in a reciprocal relationship. **On one hand, the research component (developing the predictive model and assembling data) influenced design decisions: for example, understanding the GNN's input requirements led to defining the building module geometry in a particular way (as a graph of elements and connections) and constrained how designs are parametrically modeled so that they can be interpreted by the network.**

This meant that the design of the digital workflow (the Grasshopper script and module library) was driven by research considerations about data structure and model validity. The need to achieve accurate predictions encouraged a structured representation of the building assemblies, which in turn imposed a certain modular design logic that the tool supports. **On the other hand, design considerations also influenced the research. The envisioned use-case was an architect sketching a building concept and getting environmental feedback; to accommodate this creative process, the research had to ensure the tool was interactive, fast, and visually intuitive.** Thus, design-thinking prompted

Work Package 5
Research Outcomes & Contributions

G. Research Output & Reflection
G.1 Reflection

certain research choices, such as prioritizing model speed over absolute precision and implementing graphical explanations for the impacts.

Value of the Chosen Approach and Methods:
Leveraging a GNN surrogate trained on synthetic building assemblies and embedded in a design plugin proved effective on several fronts. **GNNs naturally model the relational topology of modules and connections, capturing how changes—such as adding a floor—affect overall material use more fully than simple regressions.**

Latin Hypercube Sampling provided broad, balanced training data, enabling the model to generalize across diverse configurations. Tying volume predictions to a structured EPD database grounded the results in standardized, manufacturer-verified environmental metrics. Integrating explainability—visualizing which modules or features drive impact—transformed raw outputs into actionable guidance, fostering user confidence. Real-time inference within Grasshopper validated the surrogate approach by eliminating lengthy LCA computations, making sustainability feedback a seamless part of early design. Domain-informed regularizations (monotonicity and sign constraints) ensured predictions remained physically sensible, preventing counterintuitive results.

While this method substantially lowers the barrier to iterative, data-driven sustainable design, its accuracy ultimately depends on the representativeness of the training set, highlighting the need for ongoing validation with real-project data.

The accuracy of the GNN is inherently limited by the training data and the assumptions built into those sample assemblies. If a designer proposes a radically different configuration outside the range of the training cases, the predictions may be extrapolations with higher uncertainty.

This is not a flaw of the concept per se, but it means the method's value is highest for designs within the scope of the data – a known trade-off in surrogate modeling.

The approach was consciously narrowed to a clear focus (embodied carbon of a timber modular system), which is appropriate for a thesis scope, but extending beyond that scope would require additional methods or data. Despite these caveats, the chosen approach and methods can be deemed successful and valuable because they resulted in a functional system that met the core objectives: providing designers with immediate, informative feedback to support low-impact design decisions.

Furthermore, this approach has scholarly value as it pioneers a novel combination of techniques in the architectural domain: **linking GNN-based machine learning with an LCA database in a visual programming environment is a unique contribution that showcases the potential of data-driven sustainable design tools.**

Academic and Societal Value (and Ethical Considerations):
Academically, this work advances sustainable design, computational tools, and applied machine learning by demonstrating how a Graph Neural Network can be combined with standardized LCA data to deliver rapid environmental analysis within the early design phase. It validates literature claims that machine-learning surrogates can streamline life-cycle assessments, and introduces domain-informed constraints (e.g. monotonic skip connections) that embed engineering principles into AI models—an innovation that enhances interpretability and sets a precedent for trustworthy AI in building technology. The curated EPD database further contributes a reusable, standardized resource for future research on timber products.

Societally, the project targets a critical leverage point: early design decisions, which can determine up to 80 % of a building's total environmental impact (Chapter A), are informed by instant carbon-footprint feedback. **By empowering architects and engineers with a real-time tool, Echo helps integrate sustainability into routine workflows, lowering the barrier to environmentally conscious design and supporting broader efforts to reduce the construction sector's embodied-carbon footprint.**

Providing designers with real-time LCA feedback empowers material-efficient and low-impact decisions. If such tools become standard, they could significantly lower the construction sector's embodied-carbon footprint, advance climate goals, and incentivize manufacturers of sustainable products by clearly demonstrating their benefits. Moreover, this approach fosters a culture of data-driven decision-making in architecture, shifting focus from cost or aesthetics to life-cycle performance. **Ethical considerations—particularly ensuring prediction accuracy within validated bounds and transparently communicating underlying assumptions—have been central to the tool's development.**

By prioritizing explainability, the tool positions AI as a supportive guide, preserving the designer's agency and responsibility.**The use of verified EPD data also speaks to ethical practice: these declarations are third-party validated and follow international standards, ensuring that the source information is trustworthy and transparently documented.**

The tool is explicitly framed as decision support, not an unquestionable authority; documented scope and limitations **caution against applying it to novel cases beyond its training.** By augmenting rather than replacing human judgment, the system adheres to ethical AI principles—transparency, accountability, and competence—making assumptions and uncertainties clear and fostering trust. In sum, the project marries methodological innovation with practical impact on sustainable building and demonstrates strong potential for adaptation to other platforms, materials, and performance metrics.

The graph-based ML framework is broadly transferable: by generating new training datasets and retraining the GNN on EPD data for other materials or structural systems, the model can support performance predictions well beyond modular timber—extending, for example, to steel-frame or precast concrete assemblies.

The EPD integration strategy can be expanded to larger LCA repositories (such as EC3 or national databases), and the Grasshopper plugin demonstrates minimal adoption friction; packaging and documentation will further ease deployment, while the same core algorithms can be embedded in alternative CAD, BIM, or web-based platforms. The use of monotonicity constraints and sign-regularization provides a practical template for embedding engineering common-sense into AI models across Building Technology applications. Together, these steps will yield a fully tested, well-documented tool and a scalable framework ready for wider academic and industry use.

Work Package 5
Research Outcomes & Contributions

G. Research Output & Reflection
G.1 Reflection
Reflection Questions

1. How might the introduction of real-time, explainable GNN-driven LCA feedback alter the balance between creative intuition and data-informed decision-making in early-stage architectural design?

This question looks at what happens when we add instant, easy-to-understand carbon feedback to the very first steps of a design project. It asks whether seeing live carbon numbers helps architects try new ideas or holds them back, and how the tool can guide rather than replace their own judgement. By pointing out this balance, the question helps us think more clearly about how such tools might change the culture and thinking behind design work.

2.To what extent can the Echo methodology—linking graph-based surrogate models with standardized EPD data—be generalized to other material systems, performance metrics, and design platforms without sacrificing predictive reliability?

The second question asks how the research can stay useful over time. It invites students to judge whether a mix of graph models and standard EPD data could also serve other materials, performance goals, or software tools. Students must weigh what changes in data, limits, and checking would be needed, and how to keep the method clear while still giving trustworthy results in new settings. This view shows the project not as a single fix but as a flexible guide that can spark future work in computational building technology.

G.2 Revisting The Research Problem

The primary research problem identified in Chapter A was the lack of an integrated, explainable life-cycle assessment (LCA) mechanism at the conceptual design stage. Three specific gaps were outlined:

1.Absence of front-loaded LCA metrics
2.Delayed, post-hoc feedback
3.Opacity of predictive tools

Echo addresses each of these gaps directly:

Embedding LCA metrics in early abstractions.
By coupling a Graph Neural Network (GNN) with a consolidated EPD database, Echo computes embodied-carbon and energy metrics from the very first 3D box-and-curve sketches. Material volumes are predicted instantly and converted to LCA indicators, ensuring that every massing decision is immediately quantified against real-world environmental data.
Real-time propagation of impacts.
Echo's Grasshopper plugin reconstructs the building graph and runs inference in milliseconds. Designers see updated cumulative and per-module LCA results the moment they adjust module dimensions, opening placements, or adjacency—eliminating the weeks-long lag of traditional LCA workflows.

Transparent, actionable feedback.
Integrated Gradients and monotonic skip connections enforce physically consistent behavior (e.g., openings always reduce volume). Feature-attribution heatmaps highlight which modules or design parameters drive the impact metrics. This explainability transforms Echo from a "black box" into a trusted advisor, allowing architects to understand why a particular design scores as it does.

Unified data flow.
A single EPD lookup table harmonizes manufacturer inventories, transport factors, and end-of-life scenarios within one tool. By avoiding siloed spreadsheets and disparate databases, Echo ensures consistent, auditable LCA calculations across all design iterations.

Scalable, generalizable framework.
The graph-based GNN architecture and modular plugin design mean that Echo can be retrained for other materials or performance criteria and embedded in alternative CAD/BIM environments without fundamental reengineering.

In sum, Echo closes the loop between concept sketching and sustainability analysis by embedding robust, real-time LCA feedback directly into the architect's earliest design moves—precisely the transformation the original research problem demanded.

G.3 Recognised Limitations

Even though Echo is a strong tool, it still has limits that users should remember.

1. Some of the math is still a black box
Explainability tools such as attention maps and gradient checks let us see which input parts matter, yet the main network still turns large graphs into single numbers with learned weights and activation functions we cannot fully read. If we ever wanted to showcase step by step every single math calculation that takes place "under the hood" this would be incredibly insufficient and time consuming while at the same it would required specialised knowledge.

2. Echo assumes perfect factory and data conditions
The tool trusts that the manufacturing steps, material details, and end of life plans in the EPD files are carried out exactly as written. In real projects, factory efficiency, material sources, transport routes, and record keeping often change. When this happens, the true carbon impact can be higher than Echo shows. Since Echo cannot spot these gaps, its feedback is only as good as the real practices and data that follow the design stage.

3.Echo only covers the first design steps
Echo looks at early tasks such as forming the basic shape and estimating material volumes. It does not follow the project into later phases when engineers add detailed structure, thermal modeling, or indoor-air planning. As a result, the tool reports just part of the building's full life-cycle impact. Designers need other analyses for later changes like stronger frames, refined facades, or on-site waste handling.

H. Bibliography

1.Abed, J., Rayburg, S., Rodwell, J., & Neave, M. (2022). A Review of the Performance and Benefits of Mass Timber as an Alternative to Concrete and Steel for Improving the Sustainability of Structures. Sustainability, 14(9), 5570. https://doi.org/10.3390/su14095570

2.Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access, 6, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

3.Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., … Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks (No. arXiv:1806.01261). arXiv. https://doi.org/10.48550/arXiv.1806.01261

4.Bergman, R. D., & Alanya-Rosenbaum, S. (2017). Cradle-to-Gate Life-Cycle Assessment of Laminated Veneer Lumber Production in the United States*. Forest Products Journal, 67(5–6), 343–354. https://doi.org/10.13073/FPJ-D-16-00046

5.Bodansky, D. (2016). The Paris Climate Change Agreement: A New Hope? American Journal of International Law, 110(2), 288–319. https://doi.org/10.5305/amerjintelaw.110.2.0288

6.Calisto Friant, M., Vermeulen, W. J. V., & Salomone, R. (2021). Analysing European Union circular economy policies: Words versus actions. Sustainable Production and Consumption, 27, 337–353. https://doi.org/10.1016/j.spc.2020.11.001

7.Del Borghi, A. (2013). LCA and communication: Environmental Product Declaration. The International Journal of Life Cycle Assessment, 18(2), 293–295. https://doi.org/10.1007/s11367-012-0513-9

8.Dorst, K., & Cross, N. (2001). Creativity in the design process: Co-evolution of problem–solution. Design Studies, 22(5), 425–437. https://doi.org/10.1016/S0142-694X(01)00009-6

9.Doshi-Velez, F., & Kim, B. (2017a). Towards A Rigorous Science of Interpretable Machine Learning (No. arXiv:1702.08608). arXiv. https://doi.org/10.48550/arXiv.1702.08608

10.Doshi-Velez, F., & Kim, B. (2017b). Towards A Rigorous Science of Interpretable Machine Learning (No. arXiv:1702.08608). arXiv. https://doi.org/10.48550/arXiv.1702.08608

11.Dwivedi, V. P., & Bresson, X. (2020). A Generalization of Transformer Networks to Graphs (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2012.09699

12.Dym, C. L., Agogino, A. M., Eris, O., Frey, D. D., & Leifer, L. J. (2005). Engineering Design Thinking, Teaching, and Learning. Journal of Engineering Education, 94(1), 103–120. https://doi.org/10.1002/j.2168-9830.2005.tb00832.x

13.Eastman, C. M., & Eastman, C. M. (Eds.). (2008). BIM handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors. Wiley.

14.European Commission. Joint Research Centre. (2018a). Model for Life Cycle Assessment (LCA) of buildings. Publications Office. https://data.europa.eu/doi/10.2760/10016

15.European Commission. Joint Research Centre. (2018b). Recommendations and guidelines to foster sustainable design: EFIResources : resource efficient construction towards sustainable design. Publications Office. https://data.europa.eu/doi/10.2760/285151

16.Ferrari, S., Zoghi, M., Blázquez, T., & Dall'O', G. (2022). New Level(s) framework: Assessing the affinity between the main international Green Building Rating Systems and the european scheme. Renewable and Sustainable Energy Reviews, 155, 111924. https://doi.org/10.1016/j.rser.2021.111924

17.Finnveden, G., Hauschild, M. Z., Ekvall, T., Guinée, J., Heijungs, R., Hellweg, S., Koehler, A., Pennington, D., & Suh, S. (2009). Recent developments in Life Cycle Assessment. Journal of Environmental Management, 91(1), 1–21. https://doi.org/10.1016/j.jenvman.2009.06.018

18.Finnveden, G., & Potting, J. (2014). Life Cycle Assessment. In Encyclopedia of Toxicology (pp. 74–77). Elsevier. https://doi.org/10.1016/B978-0-12-386454-3.00627-8

H. Bibliography

19.Geissdoerfer, M., Savaget, P., Bocken, N. M. P., & Hultink, E. J. (2017). The Circular Economy – A new sustainability paradigm? Journal of Cleaner Production, 143, 757–768. https://doi.org/10.1016/j.jclepro.2016.12.048

20.Gervasio, H., Dimova, S., & European Commission (Eds.). (2018). Environmental benchmarks for buildings: EFIResources: resource efficient construction towards sustainable design. Publications Office. https://doi.org/10.2760/90028

21.Getuli, V., Capone, P., Bruttini, A., & Isaac, S. (2020). BIM-based immersive Virtual Reality for construction work space planning: A safety-oriented approach. Automation in Construction, 114, 103160. https://doi.org/10.1016/j.autcon.2020.103160

22.Ghisellini, P., Cialani, C., & Ulgiati, S. (2016). A review on circular economy: The expected transition to a balanced interplay of environmental and economic systems. Journal of Cleaner Production, 114, 11–32. https://doi.org/10.1016/j.jclepro.2015.09.007

23.Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural Message Passing for Quantum Chemistry (No. arXiv:1704.01212). arXiv. https://doi.org/10.48550/arXiv.1704.01212

24.Hamilton, W. L. (2020). Graph representation learning. Morgan & Claypool Publishers.

25.Huber, P. J. (1964). Robust Estimation of a Location Parameter. The Annals of Mathematical Statistics, 35(1), 73–101. https://doi.org/10.1214/aoms/1177703732

26.Huijbregts, M. A. J., Steinmann, Z. J. N., Elshout, P. M. F., Stam, G., Verones, F., Vieira, M., Zijp, M., Hollander, A., & Van Zelm, R. (2017). ReCiPe2016: A harmonised life cycle impact assessment method at midpoint and endpoint level. The International Journal of Life Cycle Assessment, 22(2), 138–147. https://doi.org/10.1007/s11367-016-1246-y

27.Jiang, S., Feng, F., Chen, W., Li, X., & He, X. (2021). Structure-enhanced meta-learning for few-shot graph classification. AI Open, 2, 160–167. https://doi.org/10.1016/j.aiopen.2021.08.001

28.Kim, K.-G. (2025). LCA and Consumption/Needs-Based GHG Accounting for Climate Action: A Pathway to Carbon Neutrality. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-75468-5

29.Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization (No. arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

30.Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks (No. arXiv:1609.02907). arXiv. https://doi.org/10.48550/arXiv.1609.02907

31.Kirchherr, J., Reike, D., & Hekkert, M. (2017). Conceptualizing the circular economy: An analysis of 114 definitions. Resources, Conservation and Recycling, 127, 221–232. https://doi.org/10.1016/j.resconrec.2017.09.005

32.Kömürcü, D., & Edis, E. (2025). Machine Learning Modeling for Building Energy Performance Prediction Based on Simulation Data: A Systematic Review of the Processes, Performances, and Correlation of Process-Related Variables. Buildings, 15(8), 1301. https://doi.org/10.3390/buildings15081301

33.LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

34.Leising, E., Quist, J., & Bocken, N. (2018). Circular Economy in the building sector: Three cases and a collaboration tool. Journal of Cleaner Production, 176, 976–989. https://doi.org/10.1016/j.jclepro.2017.12.010

35.Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization (No. arXiv:1711.05101). arXiv. https://doi.org/10.48550/arXiv.1711.05101

36.Ma, L., Azari, R., & Elnimeiri, M. (2024). A Building Information Modeling-Based Life Cycle Assessment of the Embodied Carbon and Environmental Impacts of High-Rise Building Structures: A Case Study. Sustainability, 16(2), 569. https://doi.org/10.3390/su16020569

37.Makridakis, S. (1993). Accuracy measures: Theoretical and practical concerns. International Journal of Forecasting, 9(4), 527–529. https://doi.org/10.1016/0169-2070(93)90079-3

38.McConville, J. R., Kvarnström, E., Jönsson, H., Kärrman, E., & Johansson, M. (2017). Source separation: Challenges & opportunities for transition in the swedish wastewater sector. Resources, Conservation and Recycling, 120, 144–156. https://doi.org/10.1016/j.resconrec.2016.12.004

39.Molnar, C. (2022). Interpretable machine learning: A guide for making black box models explainable (Second edition). Christoph Molnar.

40.Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. Digital Signal Processing, 73, 1–15. https://doi.org/10.1016/j.dsp.2017.10.011

41.Özerol, G., & Arslan Selçuk, S. (2023). Machine learning in the discipline of architecture: A review on the research trends between 2014 and 2020. International Journal of Architectural Computing, 21(1), 23–41. https://doi.org/10.1177/14780771221100102

42.Pomponi, F., & Moncaster, A. (2017). Circular economy for the built environment: A research framework. Journal of Cleaner Production, 143, 710–718. https://doi.org/10.1016/j.jclepro.2016.12.055

43.Rahla, K. M., Mateus, R., & Bragança, L. (2021). Selection Criteria for Building Materials and Components in Line with the Circular Economy Principles in the Built Environment—A Review of Current Trends. Infrastructures, 6(4), 49. https://doi.org/10.3390/infrastructures6040049

44.Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. The Annals of Mathematical Statistics, 22(3), 400–407. https://doi.org/10.1214/aoms/1177729586

45.Ruder, S. (2017). An overview of gradient descent optimization algorithms (No. arXiv:1609.04747). arXiv. https://doi.org/10.48550/arXiv.1609.04747

46.Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., & Müller, K.-R. (Eds.). (2019). Explainable AI: Interpreting, explaining and visualizing deep learning. NIPS Workshop Interpreting, Explaining and Visualizing Deep Learning ... now what?, Cham, Switzerland. Springer.

47.Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. In A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, & M. Alam (Eds.), The Semantic Web (Vol. 10843, pp. 593–607). Springer International Publishing. https://doi.org/10.1007/978-3-319-93417-4_38

48.Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

49.Song, R., Keller, A. A., & Suh, S. (2017). Rapid Life-Cycle Impact Screening Using Artificial Neural Networks. Environmental Science & Technology, 51(18), 10777–10785. https://doi.org/10.1021/acs.est.7b02862

50.Spanish Abstracts Journal of Industrial Ecology Volume 21, Number 5. (2017). Journal of Industrial Ecology, 21(5), 1412–1439. https://doi.org/10.1111/jiec.12707

51.Summa, M. G., Bottou, L., Goldfarb, B., Murtagh, F., Pardoux, C., & Touati, M. (Eds.). (2011). Large-Scale Machine Learning with Stochastic Gradient Descent Léon Bottou. In Statistical Learning and Data Science (0 ed., pp. 33–42). Chapman and Hall/CRC. https://doi.org/10.1201/b11429-6

52.Understanding Epochs. (n.d.). https://deepai.org/machine-learning-glossary-and-terms/epoch?utm_source=chatgpt.com

53.Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks (No. arXiv:1710.10903). arXiv. https://doi.org/10.48550/arXiv.1710.10903

54.Wen, X., Lv, Y., Liu, Z., Ding, Z., Lei, X., Tan, Q., & Sun, Y. (2021). Operation chart optimization of multi-hydropower system incorporating the long- and short-term fish habitat requirements. Journal of Cleaner Production, 281, 125292. https://doi.org/10.1016/j.jclepro.2020.125292

55.Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

56.Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., & Leskovec, J. (2019). Hierarchical Graph Representation Learning with Differentiable Pooling (No. arXiv:1806.08804). arXiv. https://doi.org/10.48550/arXiv.1806.08804

I. Appendix

I.1 Environmetal Product Declarations

Below you can find some of the data tables from the selected manufacturers for the research. The EPDs are also publicly available in EPD International.

Stora Enso [GWPtotal, GWPfossil, GWPbiogenic,PERT,PENRT]

Mandatory impact category indicators according to EN 15804+A2

Results per declared unit – 1 m³ LVL or LVL G by Stora Enso													
Indicator	**Unit**	**A1**	**A2**	**A3**	**A1-A3**	**A4**	**A5**	**B1-B7**	**C1**	**C2**	**C3**	**C4**	**D**
GWP-fossil	kg CO_2 eq.	7,54E+1	1,87E+1	2,64E+1	**1,21E+2**	2,72E+1	6,08E+0	0,00E+0	4,01E+0	2,22E+0	1,51E+2	0,00E+0	-3,30E+2
GWP-biogenic	kg CO_2 eq.	-8,16E+2	1,21E-2	5,76E-1	**-8,15E+2**	1,08E-2	7,50E-4	0,00E+0	6,98E-4	8,81E-4	8,17E+2	0,00E+0	-9,26E-1
GWP luluc	kg CO_2 eq.	1,37E+0	1,04E-2	4,71E-2	**1,43E+0**	1,02E-2	4,11E-4	0,00E+0	3,97E-4	8,32E-4	4,91E-3	0,00E+0	-3,42E-1
GWP total	kg CO_2 eq.	-7,39E+2	1,88E+1	2,70E+1	**-6,93E+2**	2,73E+1	6,08E+0	0,00E+0	4,01E+0	2,22E+0	9,68E+2	0,00E+0	-3,31E+2
ODP	kg CFC 11 eq.	1,00E-5	4,90E-6	2,48E-6	**1,74E-5**	6,79E-6	8,55E-7	0,00E+0	8,49E-7	5,53E-7	2,73E-6	0,00E+0	-3,53E-5
AP	mol H^+ eq.	5,00E-1	7,61E-2	3,05E-1	**8,81E-1**	8,68E-2	2,04E-2	0,00E+0	2,00E-2	7,07E-3	2,46E-1	0,00E+0	-9,16E-1
EP-freshwater	kg P eq.	7,86E-3	2,23E-4	2,04E-3	**1,01E-2**	1,94E-4	1,35E-5	0,00E+0	1,32E-5	1,58E-5	1,84E-4	0,00E+0	-1,54E-2
EP-marine	kg N eq.	1,65E-1	1,90E-2	9,59E-2	**2,80E-1**	1,91E-2	7,93E-3	0,00E+0	7,75E-3	1,55E-3	1,12E-1	0,00E+0	-1,44E-1
EP-terrestrial	mol N eq.	1,45E+0	2,11E-1	1,24E+0	**2,90E+0**	2,12E-1	8,71E-2	0,00E+0	8,52E-2	1,73E-2	1,26E+0	0,00E+0	-1,63E+0
POCP	kg NMVOC eq.	3,51E-1	7,22E-2	2,44E-1	**6,67E-1**	8,36E-2	2,46E-2	0,00E+0	2,41E-2	6,81E-3	3,33E-1	0,00E+0	-4,81E-1
ADP minerals&metals[3]	kg Sb eq.	1,33E-3	5,59E-5	1,32E-4	**1,52E-3**	6,52E-5	2,15E-6	0,00E+0	2,04E-6	5,31E-6	4,17E-5	0,00E+0	-1,72E-4
ADP-fossil[3]	MJ	1,64E+3	3,34E+2	2,64E+3	**4,61E+3**	4,43E+2	5,49E+1	0,00E+0	5,45E+1	3,61E+1	1,54E+2	0,00E+0	-6,26E+3
WDP[3]	m^3	9,77E+1	1,18E+0	1,89E+1	**1,18E+2**	1,48E+0	8,51E-2	0,00E+0	7,77E-2	1,21E-1	2,71E+0	0,00E+0	-3,45E+1
Acronyms	GWP-fossil = Global Warming Potential fossil fuels; GWP-biogenic = Global Warming Potential biogenic; GWP-luluc = Global Warming Potential land use and land use change; ODP = Depletion potential of the stratospheric ozone layer; AP = Acidification potential, Accumulated Exceedance; EP-freshwater = Eutrophication potential, fraction of nutrients reaching freshwater end compartment; EP-marine = Eutrophication potential, fraction of nutrients reaching marine end compartment; EP-terrestrial = Eutrophication potential, Accumulated Exceedance; POCP = Formation potential of tropospheric ozone; ADP-minerals&metals = Abiotic depletion potential for non-fossil resources; ADP-fossil = Abiotic depletion for fossil resources potential; WDP = Water (user) deprivation potential, deprivation-weighted water consumption												

Resource use indicators

Results per declared unit – 1 m³ LVL or LVL G by Stora Enso													
Indicator	**Unit**	**A1**	**A2**	**A3**	**A1-A3**	**A4**	**A5**	**B1-B7**	**C1**	**C2**	**C3**	**C4**	**D**
PERE	MJ	8,97E+3	7,39E+0	6,19E+2	**9,60E+3**	5,64E+0	3,15E-1	0,00E+0	3,07E-1	4,59E-1	4,98E+0	0,00E+0	-5,50E+2
PERM	MJ	7,59E+3	0,00E+0	0,00E+0	**7,59E+3**	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	-7,59E+3	0,00E+0	0,00E+0
PERT	MJ	1,66E+4	7,39E+0	6,19E+2	**1,72E+4**	5,64E+0	3,15E-1	0,00E+0	3,07E-1	4,59E-1	-7,58E+3	0,00E+0	-5,50E+2
PENRE	MJ	1,77E+3	3,54E+2	2,66E+3	**4,78E+3**	4,71E+2	5,84E+1	0,00E+0	5,79E+1	3,83E+1	1,65E+2	0,00E+0	-6,76E+3
PENRM	MJ	1,91E+3	0,00E+0	5,45E+1	**1,96E+3**	0,00E+0	-5,45E+1	0,00E+0	0,00E+0	0,00E+0	-1,91E+3	0,00E+0	0,00E+0
PENRT	MJ	3,68E+3	3,54E+2	2,72E+3	**6,75E+3**	4,71E+2	3,83E+0	0,00E+0	5,79E+1	3,83E+1	-1,74E+3	0,00E+0	-6,76E+3
SM	kg	0,00E+0	0,00E+0	0,00E+0	**0,00E+0**	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
RSF	MJ	0,00E+0	0,00E+0	0,00E+0	**0,00E+0**	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
NRSF	MJ	0,00E+0	0,00E+0	0,00E+0	**0,00E+0**	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
FW	m^3	4,23E+0	4,81E-2	8,18E-1	**5,10E+0**	4,88E-2	3,02E-3	0,00E+0	2,75E-3	3,97E-3	2,94E-1	0,00E+0	-2,64E+0
Acronyms	PERE = Use of renewable primary energy excluding renewable primary energy resources used as raw materials; PERM = Use of renewable primary energy resources used as raw materials; PERT = Total use of renewable primary energy resources; PENRE = Use of non-renewable primary energy excluding non-renewable primary energy resources used as raw materials; PENRM = Use of non-renewable primary energy resources used as raw materials; PENRT = Total use of non-renewable primary energy re-sources; SM = Use of secondary material; RSF = Use of renewable secondary fuels; NRSF = Use of non-renewable secondary fuels; FW = Use of net fresh water												

Figure 85: EPD Stora Enso
Source: All of the documentation can be found in https://www.environdec.com/home

I. Appendix
I.1 Environmetal Product Declarations

Stora Enso [EOL Scenarios]

Results per declared unit – 1 m³ LVL or LVL G by Stora Enso																
		Re-use					Recycling					Landfill				
Indicator	**Unit**	**C1**	**C2**	**C3**	**C4**	**D**	**C1**	**C2**	**C3**	**C4**	**D**	**C1**	**C2**	**C3**	**C4**	**D**
GWP-fossil	kg CO_2 eq.	4,01E+0	2,22E+0	0,00E+0	0,00E+0	-1,14E+2	4,01E+0	2,22E+0	5,99E+0	0,00E+0	-1,72E+1	4,01E+0	2,22E+0	0,00E+0	1,07E+1	-4,79E-2
GWP-biogenic	kg CO_2 eq.	6,98E-4	8,81E-4	8,17E+2	0,00E+0	-1,20E+0	6,98E-4	8,81E-4	8,17E+2	0,00E+0	-1,77E-1	6,98E-4	8,81E-4	0,00E+0	1,08E+3	-1,59E-4
GWP luluc	kg CO_2 eq.	3,97E-4	8,32E-4	0,00E+0	0,00E+0	-1,43E+0	3,97E-4	8,32E-4	5,98E-4	0,00E+0	-1,97E-1	3,97E-4	8,32E-4	0,00E+0	1,14E-3	-5,88E-5
GWP total	kg CO_2 eq.	4,01E+0	2,22E+0	8,17E+2	0,00E+0	-1,17E+2	4,01E+0	2,22E+0	8,23E+2	0,00E+0	-1,76E+1	4,01E+0	2,22E+0	0,00E+0	1,09E+3	-4,81E-2
ODP	kg CFC 11 eq.	8,49E-7	5,53E-7	0,00E+0	0,00E+0	-1,61E-5	8,49E-7	5,53E-7	1,28E-6	0,00E+0	-1,22E-6	8,49E-7	5,53E-7	0,00E+0	1,67E-6	-4,74E-9
AP	mol H^+ eq.	2,00E-2	7,07E-3	0,00E+0	0,00E+0	-8,19E-1	2,00E-2	7,07E-3	6,22E-2	0,00E+0	-1,45E-1	2,00E-2	7,07E-3	0,00E+0	3,94E-2	-1,51E-4
EP-freshwater	kg P eq.	1,32E-5	1,58E-5	0,00E+0	0,00E+0	-1,01E-2	1,32E-5	1,58E-5	1,99E-5	0,00E+0	-2,54E-3	1,32E-5	1,58E-5	0,00E+0	5,45E-5	-2,65E-6
EP-marine	kg N eq.	7,75E-3	1,55E-3	0,00E+0	0,00E+0	-2,52E-1	7,75E-3	1,55E-3	2,75E-2	0,00E+0	-3,75E-2	7,75E-3	1,55E-3	0,00E+0	3,56E-2	-2,28E-5
EP-terrestrial	mol N eq.	8,52E-2	1,73E-2	0,00E+0	0,00E+0	-2,60E+0	8,52E-2	1,73E-2	3,02E-1	0,00E+0	-4,36E-1	8,52E-2	1,73E-2	0,00E+0	1,62E-1	-2,58E-4
POCP	kg NMVOC eq.	2,41E-2	6,81E-3	0,00E+0	0,00E+0	-5,84E-1	2,41E-2	6,81E-3	8,30E-2	0,00E+0	-1,46E-1	2,41E-2	6,81E-3	0,00E+0	1,25E-1	-7,51E-5
ADP minerals&metals[3]	kg Sb eq.	2,04E-6	5,31E-6	0,00E+0	0,00E+0	-1,52E-3	2,04E-6	5,31E-6	3,08E-6	0,00E+0	-1,68E-4	2,04E-6	5,31E-6	0,00E+0	1,54E-5	-2,83E-8
ADP-fossil[3]	MJ	5,45E+1	3,61E+1	0,00E+0	0,00E+0	-4,53E+3	5,45E+1	3,61E+1	8,21E+1	0,00E+0	-3,66E+2	5,45E+1	3,61E+1	0,00E+0	1,21E+2	-9,27E-1
WDP[3]	m^3	7,77E-2	1,21E-1	0,00E+0	0,00E+0	-1,18E+2	7,77E-2	1,21E-1	1,17E-1	0,00E+0	-2,18E+1	7,77E-2	1,21E-1	0,00E+0	5,65E-1	-5,91E-3
GWP-GHG[4]	kg CO_2 eq.	4,01E+0	2,22E+0	0,00E+0	0,00E+0	-1,16E+2	4,01E+0	2,22E+0	5,99E+0	0,00E+0	-1,74E+1	4,01E+0	2,22E+0	0,00E+0	2,94E+2	-4,80E-2
PERE	MJ	3,07E-1	4,59E-1	0,00E+0	0,00E+0	-9,60E+3	3,07E-1	4,59E-1	4,62E-1	0,00E+0	-3,17E+3	3,07E-1	4,59E-1	0,00E+0	5,47E+0	-9,46E-2
PERM	MJ	0,00E+0	0,00E+0	-7,59E+3	0,00E+0	-7,59E+3	0,00E+0	0,00E+0	-7,59E+3	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	-7,59E+3	0,00E+0
PERT	MJ	3,07E-1	4,59E-1	-7,59E+3	0,00E+0	-1,72E+4	3,07E-1	4,59E-1	-7,59E+3	0,00E+0	-3,17E+3	3,07E-1	4,59E-1	0,00E+0	-7,58E+3	-9,46E-2
PENRE	MJ	5,79E+1	3,83E+1	0,00E+0	0,00E+0	-4,70E+3	5,79E+1	3,83E+1	8,72E+1	0,00E+0	-3,91E+2	5,79E+1	3,83E+1	0,00E+0	1,28E+2	-9,96E-1
PENRM	MJ	0,00E+0	0,00E+0	-1,91E+3	0,00E+0	-1,96E+3	0,00E+0	0,00E+0	-1,91E+3	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	-1,91E+3	0,00E+0
PENRT	MJ	5,79E+1	3,83E+1	-1,91E+3	0,00E+0	-6,66E+3	5,79E+1	3,83E+1	-1,82E+3	0,00E+0	-3,91E+2	5,79E+1	3,83E+1	0,00E+0	-1,78E+3	-9,96E-1
SM	kg	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
RSF	MJ	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
NRSF	MJ	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
FW	m^3	2,75E-3	3,97E-3	0,00E+0	0,00E+0	-5,09E+0	2,75E-3	3,97E-3	4,14E-3	0,00E+0	-6,74E-1	2,75E-3	3,97E-3	0,00E+0	1,52E-1	-4,53E-4
HWD	kg	1,49E-4	8,74E-5	0,00E+0	0,00E+0	-8,19E-1	1,49E-4	8,74E-5	2,25E-4	0,00E+0	-4,28E-4	1,49E-4	8,74E-5	0,00E+0	1,47E-4	-6,95E-7
NHWD	kg	7,28E-2	3,37E+0	0,00E+0	0,00E+0	-3,79E+1	7,28E-2	3,37E+0	1,10E-1	0,00E+0	-3,91E+0	7,28E-2	3,37E+0	0,00E+0	1,05E+3	-1,86E-3
RWD	kg	3,76E-4	2,44E-4	0,00E+0	0,00E+0	-4,47E-2	3,76E-4	2,44E-4	5,67E-4	0,00E+0	-1,65E-3	3,76E-4	2,44E-4	0,00E+0	7,81E-4	-4,12E-6
CRU	kg	0,00E+0	0,00E+0	5,10E+2	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
MFR	kg	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	5,10E+2	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
MER	kg	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0
EEE	MJ	0,00E+0	0,00E+0	0,00E+0	0,00E+0	-1,53E+1	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	2,29E-1	0,00E+0
EET	MJ	0,00E+0	0,00E+0	0,00E+0	0,00E+0	-3,72E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	0,00E+0	4,72E-1	0,00E+0
Acronyms	GWP-fossil = Global Warming Potential fossil fuels; GWP-biogenic = Global Warming Potential biogenic; GWP-luluc = Global Warming Potential land use and land use change; ODP = Depletion potential of the stratospheric ozone layer; AP = Acidification potential, Accumulated Exceedance; EP-freshwater = Eutrophication potential, fraction of nutrients reaching freshwater end compartment; EP-marine = Eutrophication potential, fraction of nutrients reaching marine end compartment; EP-terrestrial = Eutrophication potential, Accumulated Exceedance; POCP = Formation potential of tropospheric ozone; ADP-minerals&metals = Abiotic depletion potential for non-fossil resources; ADP-fossil = Abiotic depletion for fossil resources potential; WDP = Water (user) deprivation potential, deprivation-weighted water consumption; PERE = Use of renewable primary energy excluding renewable primary energy resources used as raw materials; PERM = Use of renewable primary energy resources used as raw materials; PERT = Total use of renewable primary energy resources; PENRE = Use of non-renewable primary energy excluding non-renewable primary energy resources used as raw materials; PENRM = Use of non-renewable primary energy resources used as raw materials; PENRT = Total use of non-renewable primary energy re-sources; SM = Use of secondary material; RSF = Use of renewable secondary fuels; NRSF = Use of non-renewable secondary fuels; FW = Use of net fresh water; HWD = Hazardous waste; NHWD = Non-hazardous waste; RWD = Radioactive waste; CRU = Components for re-use; MFR = Materials for recycling; MER = Materials for energy recovery; EEE = Exported energy, electric; EET = Exported energy, thermal; *															

Figure 86: EPD Stora Enso
Source: All of the documentation can be found in https://www.environdec.com/home

I. Appendix

I.1 Environmetal Product Declarations

Metsa [GWPtotal, GWPfossil, GWPbiogenic,PERT,PENRT]

Indicator	Unit	A1-A3	A4	A5
Global Warming Potential - total (GWP-total)*	kg CO_2 eq.	-382	19.8	37.5
Global Warming Potential - fossil fuels (GWP-fossil)	kg CO_2 eq.	280	19.7	20.3
Global Warming Potential - biogenic (GWP-biogenic)*	kg CO_2 eq.	-663	$5.14 \cdot 10^{-2}$	17.1

End Of Life Scenarios

SCENARIO 1: INCINERATION AS SECONDARY FUEL

Indicator	Unit	C1	C2	C3	C4	D
Global Warming Potential - total (GWP-total)	kg CO_2 eq.	$2.00 \cdot 10^{-2}$	1.70	785	0	-330
Global Warming Potential - fossil fuels (GWP-fossil)	kg CO_2 eq.	$1.99 \cdot 10^{-2}$	1.67	0.830	0	-422
Global Warming Potential - biogenic (GWP-biogenic)	kg CO_2 eq.	$6.62 \cdot 10^{-5}$	$1.25 \cdot 10^{-2}$	784	0	92.9

SCENARIO 2: RECYCLING

Indicator	Unit	C1	C2	C3	C4	D
Global Warming Potential - total (GWP-total)	kg CO_2 eq.	$2.00 \cdot 10^{-2}$	1.70	785	0	809
Global Warming Potential - fossil fuels (GWP-fossil)	kg CO_2 eq.	$1.99 \cdot 10^{-2}$	1.67	0.830	0	-46.5
Global Warming Potential - biogenic (GWP-biogenic)	kg CO_2 eq.	$6.62 \cdot 10^{-5}$	$1.25 \cdot 10^{-2}$	784	0	856

Figure 87: Metsa EPD
Source: All of the documentation can be found in https://www.environdec.com/home

VMG Lignum [GWPtotal, GWPfossil, GWPbiogenic,PERT,PENRT]

MANDATORY IMPACT CATEGORY INDICATORS ACCORDING TO EN 15804+A2

Impact category	Unit	A1-A3	A4	C1	C2	C3	C4	D
GWP-total	kg CO_2e	-7,35E+02	3,01E+01	4,95E-02	2,77E+00	1,05E+03	0,00E+00	-1,61E+02
GWP-fossil	kg CO_2e	3,00E+02	3,04E+01	4,94E-02	2,77E+00	8,86E+03	0,00E+00	-1,61E+02
GWP-biogenic	kg CO_2e	-1,04E+03	2,01E-02	1,38E-05	2,01E-03	1,04E+03	0,00E+00	-2,72E-02
GWP-luluc	kg CO_2e	1,24E+00	9,79E-03	4,18E-06	8,34E-04	1,72E-03	0,00E+00	-3,69E-03
ODP	kg $CFC_{-11}e$	2,36E-05	7,07E-06	1,07E-08	6,52E-07	7,12E-07	0,00E+00	-2,37E-05
AP	mol H^+e	1,53E+00	1,87E-01	5,17E-04	1,16E-02	9,99E-02	0,00E+00	-1,48E-01
EP-freshwater	kg Pe	1,87E-02	2,39E-04	2,00E-07	2,25E-05	1,18E-04	0,00E+00	-1,30E-04
EP-marine	kg Ne	3,78E-01	5,27E-02	2,28E-04	3,51E-03	4,73E-02	0,00E+00	-4,53E-02
EP-terrestrial	mol Ne	4,17E+00	5,83E-01	2,51E-03	3,87E-02	5,03E-01	0,00E+00	-4,96E-01
POCP	kg NMVOCe	1,87E+00	1,76E-01	6,89E-04	1,25E-02	1,24E-01	0,00E+00	-1,65E-01
ADP- minerals &metals*	kg Sbe	4,14E-03	4,97E-04	7,55E-08	4,73E-05	1,17E-04	0,00E+00	-4,32E-05
ADP-fossil*	MJ	6,51E+03	4,67E+02	6,81E-01	4,31E+01	7,78E+01	0,00E+00	-2,68E+03
WDP*	m^3e depr.	5,10E+03	1,69E+02	1,27E-01	1,60E+01	-6,88E+00	0,00E+00	-2,35E+01

RESOURCE USE INDICATORS

Impact category	Unit	A1-A3	A4	C1	C2	C3	C4	D
PERE	MJ	4,78E+03	5,71E+00	3,68E-03	5,43E+01	9,25E+02	0,00E+00	-4,03E+00
PERM	MJ	4,65E+03	0,00E+00	0,00E+00	0,00E+00	1,46E+04	0,00E+00	-1,46E+04
PERT	MJ	5,13E+04	5,71E+00	3,68E-03	5,43E-01	1,55E+04	0,00E+00	-1,46E+04
PENRE	MJ	6,47E+03	4,67E+02	6,81E-01	4,31E+01	2,99E+03	0,00E+00	-2,68E+03
PENRM	MJ	1,61E+03	0,00E+00	0,00E+00	0,00E+00	8,23E+02	0,00E+00	-8,23E+02
PENRT	MJ	8,09E+03	4,67E+02	6,81E-01	4,31E+01	3,82E+03	0,00E+00	-3,50E+03

Figure 88: VMG Lignum EPD
Source: All of the documentation can be found in https://www.environdec.com/home

I. Appendix
I.2 Code Developed For The Research
I.2.1 The Graph Neural Network

```
import os
import glob
import json
import time
import random
import logging

import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.autograd
import dgl
from dgl.nn import GraphConv
from dgl.dataloading import GraphDataLoader
import matplotlib.pyplot as plt
import seaborn as sns

# ==========================
# 1. Configuration & Flags
# ==========================
# Whether to run training; if False, just loads the saved model for inference
TRAIN = True

# Visualization toggles:
# - ENABLE_VISUALIZATIONS: whether to produce any plots at all
# - PLOT_VISUALIZATIONS: if True, show plots interactively; if False, save them to disk
ENABLE_VISUALIZATIONS = True
PLOT_VISUALIZATIONS    = False  # When false, figures are saved under VIZ_FOLDER

# Paths to dataset and output folders
JSON_FOLDER              = "/home/simospc/projectEcho/dataset/training_validation"
EXTRA_VALIDATION_FOLDER  = "/home/simospc/projectEcho/dataset/EXTRA_VALIDATION_FOLDER"
EXTRA_GROUND_TRUTH_FOLDER = "/home/simospc/projectEcho/dataset/EXTRA_GROUND_TRUTH_FOLDER"
VIZ_FOLDER               = "/home/simospc/projectEcho/viz_folder"
MODEL_SAVE_FOLDER        = "/home/simospc/projectEcho"

# Training hyperparameters
BATCH_SIZE    = 8
NUM_EPOCHS    = 500
LEARNING_RATE = 5e-5
TRAIN_SPLIT   = 0.65  # Fraction of in-distribution data to use for training
SEED          = 42

# Device selection
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Feature names for input and output
INPUT_FEATURE_NAMES = [
    "length_m", "width_m", "lengthkey", "widthkey",
    "DoorCount", "WindowCount", "OpeningArea",
    "AreaAfterOpenings", "SideIntersectionArea",
    "Level", "total_modules"
]
OUTPUT_FEATURE_NAMES = ["ObjectCount", "ObjectVolume"]

# Derived dimensions
```



```
IN_FEATS      = len(INPUT_FEATURE_NAMES)
HIDDEN_FEATS  = 32
OUT_FEATS     = len(OUTPUT_FEATURE_NAMES)
DROPOUT_RATE  = 0.35  # Dropout probability

# Floating-point display options
np.set_printoptions(suppress=True, formatter={'float_kind': lambda x: f"{x:.4f}"})

# Logging setup
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(name)s: %(message)s",
    datefmt="%Y-%m-%d %H:%M:%S"
)
logger = logging.getLogger(__name__)

# ==========================
# 2. Visualization Helpers
# ==========================
CURRENT_SESSION_FOLDER = None

def get_new_viz_session_folder(base_folder: str) -> str:
    """
    Create a new subfolder for this run's visualizations.
    """
    os.makedirs(base_folder, exist_ok=True)
    existing = [d for d in os.listdir(base_folder)
                if os.path.isdir(os.path.join(base_folder, d)) and d.startswith("session_")]
    nums = [int(d.split("_")[1]) for d in existing if d.split("_")[1].isdigit()]
    next_idx = max(nums, default=-1) + 1
    session_folder = os.path.join(base_folder, f"session_{next_idx}")
    os.makedirs(session_folder, exist_ok=True)
    return session_folder

def handle_figure(fig: plt.Figure, filename: str):
    """
    Either show or save a matplotlib figure depending on flags.
    """
    global CURRENT_SESSION_FOLDER
    if not ENABLE_VISUALIZATIONS:
        plt.close(fig)
        return
    if PLOT_VISUALIZATIONS:
        plt.show()
        plt.close(fig)
    else:
        if CURRENT_SESSION_FOLDER is None:
            try:
                CURRENT_SESSION_FOLDER = get_new_viz_session_folder(VIZ_FOLDER)
            except Exception as e:
                logger.error(f"Could not create viz session folder: {e}")
                plt.close(fig)
                return
        out_path = os.path.join(CURRENT_SESSION_FOLDER, filename)
        try:
            fig.savefig(out_path, bbox_inches='tight')
            logger.info(f"Saved figure: {out_path}")
        except Exception as e:
            logger.error(f"Failed to save figure {out_path}: {e}")
        finally:
            plt.close(fig)

# Use a clean seaborn style for all plots
sns.set_theme(style='whitegrid')
```


BIBLIOGRAPHY & APPENDIX

I. Appendix
I.2 Code Developed For The Research

```
# ==========================
# 3. Data Preparation
# ==========================
def load_json_files(json_folder: str) -> list:
    """
    Load all JSON files in a folder into Python dicts.
    """
    logger.info(f"Loading JSON files from {json_folder}")
    if not os.path.isdir(json_folder):
        logger.error(f"Folder not found: {json_folder}")
        return []
    paths = glob.glob(os.path.join(json_folder, "*.json"))
    logger.info(f"Found {len(paths)} JSON files")
    designs = []
    for path in paths:
        try:
            with open(path, 'r', encoding='utf-8') as f:
                design = json.load(f)
            did = design.get('design_id')
            if not did:
                # fallback: use filename as design_id
                did = os.path.splitext(os.path.basename(path))[0]
                design['design_id'] = did
            designs.append(design)
        except Exception as e:
            logger.error(f"Error loading {path}: {e}")
    return designs

def build_graph_from_design(design: dict) -> dgl.DGLGraph:
    """
    Convert a design dict into a DGLGraph, with node features and labels.
    """
    did   = design.get('design_id', 'Unknown')
    nodes = design.get('nodes', [])
    if not nodes:
        raise ValueError(f"Design {did} has no nodes")

    # Map NodeID strings to integer indices
    idx_map = {n['NodeID']: i for i, n in enumerate(nodes)}

    feat_list, target_list = [], []
    for n in nodes:
        # Check presence of all required fields
        missing_feats = [k for k in INPUT_FEATURE_NAMES if k not in n]
        missing_tgts  = [k for k in OUTPUT_FEATURE_NAMES if k not in n]
        if missing_feats:
            raise KeyError(f"{did}: missing features {missing_feats}")
        if missing_tgts:
            raise KeyError(f"{did}: missing targets {missing_tgts}")
        # Collect features and targets
        feat_list.append([n[k] for k in INPUT_FEATURE_NAMES])
        target_list.append([n[k] for k in OUTPUT_FEATURE_NAMES])

    # Create tensors
    feats   = torch.tensor(feat_list, dtype=torch.float32)
    targets = torch.tensor(target_list, dtype=torch.float32)
```



```
    # Build edge lists from connectivity
    src, dst = [], []
    for n in nodes:
        i = idx_map[n['NodeID']]
        for nb in n.get('connectivity', []):
            j = idx_map.get(nb)
            if j is None:
                logger.warning(f"{did}: invalid neighbor {nb}")
            elif i != j:
                src.append(i)
                dst.append(j)

    # Create DGLGraph and add self-loops
    g = dgl.graph((src, dst), num_nodes=len(nodes))
    g = dgl.add_self_loop(g)

    # Attach node data
    g.ndata['feat']  = feats
    g.ndata['label'] = targets
    g.design_id      = did
    g.node_ids       = [n['NodeID'] for n in nodes]

    # Sanity checks
    assert feats.shape[1] == IN_FEATS,  \
        f"Feature dim mismatch: {feats.shape[1]} vs {IN_FEATS}"
    assert targets.shape[1] == OUT_FEATS, \
        f"Target dim mismatch: {targets.shape[1]} vs {OUT_FEATS}"

    return g

def create_dataset(json_folder: str) -> list:
    """
    Load JSON designs and convert them to a list of DGLGraphs.
    """
    designs = load_json_files(json_folder)
    seen, graphs = set(), []
    for d in designs:
        did = d['design_id']
        if did in seen:
            continue
        try:
            g = build_graph_from_design(d)
            graphs.append(g)
            seen.add(did)
        except Exception as e:
            logger.error(f"Failed building graph {did}: {e}")
    logger.info(f"Dataset: {len(graphs)}/{len(designs)} graphs created")
    return graphs

def create_dataset_dict(json_folder: str) -> dict:
    """
    Create a dict mapping design_id to DGLGraph for quick lookup.
    """
    graphs = create_dataset(json_folder)
    dct = {g.design_id: g for g in graphs}
    logger.info(f"Graph dict with {len(dct)} entries")
    return dct

def collate_fn(samples: list) -> tuple:
    """
    Custom collate function for DGL GraphDataLoader.
    """
    valid = [g for g in samples if hasattr(g, 'design_id')]
    if not valid:
```

I. Appendix
I.2 Code Developed For The Research

```
        return None, []
    bg = dgl.batch(valid)
    return bg, [g.design_id for g in valid]

# ==========================
# 4. Model Definition
# ==========================
class GraphRegressor(nn.Module):
    """
    3-layer graph convolutional network with a skip connection and dropout.
    """
    def __init__(self,
                 in_feats=IN_FEATS,
                 hidden_feats=HIDDEN_FEATS,
                 out_feats=OUT_FEATS,
                 dropout_rate=DROPOUT_RATE):
        super().__init__()
        # Three graph-convolutional layers
        self.conv1   = GraphConv(in_feats, hidden_feats, allow_zero_in_degree=True)
        self.conv2   = GraphConv(hidden_feats, hidden_feats, allow_zero_in_degree=True)
        self.conv3   = GraphConv(hidden_feats, hidden_feats, allow_zero_in_degree=True)
        # Skip connection: project raw inputs up to hidden dimension
        self.fc_skip = nn.Linear(in_feats, hidden_feats)
        # Final linear layer to produce outputs
        self.fc_out  = nn.Linear(hidden_feats, out_feats)
        # Dropout for regularization
        self.dropout = nn.Dropout(dropout_rate)

        logger.info(
            f"Initialized GraphRegressor w/ skip & dropout: "
            f"in={in_feats}, hid={hidden_feats}, out={out_feats}, drop={dropout_rate}"
        )

    def forward(self, g, x):
        """
        Forward pass:
         1) Three graph conv + ReLU + dropout
         2) Skip connection from inputs
         3) Final linear projection
        """
        h = F.relu(self.conv1(g, x))
        h = self.dropout(h)
        h = F.relu(self.conv2(g, h))
        h = self.dropout(h)
        h = F.relu(self.conv3(g, h))
        h = self.dropout(h)

        skip = F.relu(self.fc_skip(x))
        skip = self.dropout(skip)

        # Combine and project
        return self.fc_out(h + skip)

# ==========================
# 5. Prediction & OOD Utilities
# ==========================
@torch.no_grad()
def predict(model: nn.Module, graph: dgl.DGLGraph, device: torch.device):
    """
```



```
    Run forward pass and return node-wise predictions (on CPU).
    """
    if graph is None:
        logger.error("Predict: received None graph")
        return None
    g = graph.to(device)
    return model(g, g.ndata['feat']).cpu()

@torch.no_grad()
def gather_predictions(model, dataset, device):
    """
    Collect predictions and labels for in-distribution dataset.
    """
    all_p, all_a = [], []
    for g in dataset:
        gdev = g.to(device)
        p = model(gdev, gdev.ndata['feat']).cpu().numpy()
        a = gdev.ndata['label'].cpu().numpy()
        all_p.append(p)
        all_a.append(a)
    if not all_p:
        return np.empty((0, OUT_FEATS)), np.empty((0, OUT_FEATS))
    return np.vstack(all_p), np.vstack(all_a)

@torch.no_grad()
def gather_predictions_ood(model, dataset, gt_dict, device):
    """
    Collect predictions and ground-truth for OOD dataset.
    """
    all_p, all_a = [], []
    for g in dataset:
        did = g.design_id
        if did not in gt_dict:
            logger.warning(f"OOD GT missing for {did}")
            continue
        gdev = g.to(device)
        p = model(gdev, gdev.ndata['feat']).cpu().numpy()
        a = gt_dict[did].ndata['label'].cpu().numpy()
        all_p.append(p)
        all_a.append(a)
    if not all_p:
        return np.empty((0, OUT_FEATS)), np.empty((0, OUT_FEATS))
    return np.vstack(all_p), np.vstack(all_a)

def evaluate_extra_validation(model, extra_val_dataset, gt_dict, device):
    """
    Compute MAPE, accuracy, and R^2 for OOD dataset.
    """
    p_all, t_all = gather_predictions_ood(model, extra_val_dataset, gt_dict, device)
    if p_all.size == 0:
        return None, None, None
    mape = np.mean(np.abs(t_all - p_all) / (np.abs(t_all) + 1e-6)) * 100
    acc  = 100 - mape
    t_flat, p_flat = t_all.flatten(), p_all.flatten()
    ss_res = np.sum((t_flat - p_flat)**2)
    ss_tot = np.sum((t_flat - np.mean(t_flat))**2)
    r2     = 1 - ss_res/ss_tot if ss_tot > 0 else 0.0
    return mape, acc, r2

# ==========================
# 6. Evaluation & Training
# ==========================
def evaluate_model(model, dataset, device):
    """
```

I. Appendix
I.2 Code Developed For The Research

```
    Compute overall MAPE, accuracy, and R^2 on in-distribution data.
    """
    p_all, t_all = gather_predictions(model, dataset, device)
    if p_all.size == 0:
        return 0.0, 0.0, 0.0
    mape = np.mean(np.abs(t_all - p_all) / (np.abs(t_all) + 1e-6)) * 100
    acc  = 100 - mape
    t_flat, p_flat = t_all.flatten(), p_all.flatten()
    ss_res = np.sum((t_flat - p_flat)**2)
    ss_tot = np.sum((t_flat - np.mean(t_flat))**2)
    r2     = 1 - ss_res/ss_tot if ss_tot > 0 else 0.0
    return mape, acc, r2

def train_and_evaluate(model, train_loader, val_dataset,
                       optimizer, loss_fn, device, num_epochs=NUM_EPOCHS):
    """
    Perform training with periodic validation. Uses Huber loss for robustness.
    """
    train_losses, val_accs, val_losses, val_r2s = [], [], [], []
    for epoch in range(1, num_epochs+1):
        model.train()
        epoch_loss, count = 0.0, 0
        for bg, _ in train_loader:
            gdev = bg.to(device)
            out  = model(gdev, gdev.ndata['feat'])
            loss = loss_fn(out, gdev.ndata['label'])
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            epoch_loss += loss.item()
            count += 1

        train_loss = epoch_loss / count if count else 0.0

        # Validate
        mape, acc, r2 = evaluate_model(model, val_dataset, device)
        train_losses.append(train_loss)
        val_accs.append(acc)
        val_losses.append(train_loss)
        val_r2s.append(r2)

        logger.info(
            f"Epoch {epoch}/{num_epochs} | "
            f"TrainLoss={train_loss:.4f} | "
            f"ValMAPE={mape:.2f}% | ValAcc={acc:.2f}% | ValR2={r2:.4f}"
        )

    return train_losses, val_accs, val_losses, val_r2s

# ==========================
# 7. Plotting Functions
# ==========================
def plot_training_metrics(train_losses, val_accuracies, val_losses=None):
    """
    Plot loss curves (train & val) and val accuracy, without markers.
    """
    epochs = np.arange(1, len(train_losses)+1)
    fig, ax1 = plt.subplots(figsize=(10,6))
```



```
    # Train & validation loss
    ax1.plot(epochs, train_losses, label='Train Loss')
    if val_losses is not None:
        ax1.plot(epochs, val_losses, linestyle='--', label='Val Loss')
    ax1.set_xlabel('Epoch')
    ax1.set_ylabel('Loss')
    ax1.legend(loc='upper left')

    # Validation accuracy on secondary axis
    ax2 = ax1.twinx()
    ax2.plot(epochs, val_accuracies, linestyle='-', label='Val Accuracy')
    ax2.set_ylabel('Accuracy (%)')
    ax2.legend(loc='upper right')

    fig.tight_layout()
    handle_figure(fig, 'training_metrics.png')

def plot_prediction_vs_actual_scatter(predictions, actuals,
                                      output_dim, output_name, dataset_label):
    """
    Scatter plot for predicted vs actual for a single output dimension.
    """
    if predictions.size == 0 or actuals.size == 0:
        logger.warning(f"No data for scatter {output_name}")
        return

    pred = predictions[:, output_dim]
    act  = actuals[:, output_dim]
    fig, ax = plt.subplots()
    ax.scatter(act, pred, alpha=0.6)
    mn, mx = min(act.min(), pred.min()), max(act.max(), pred.max())
    ax.plot([mn, mx], [mn, mx], 'r--', linewidth=1)
    ax.set_xlabel(f"Actual {output_name}")
    ax.set_ylabel(f"Predicted {output_name}")
    ax.set_title(f"{dataset_label}: {output_name}")
    fig.tight_layout()

    filename = f"{dataset_label}_{output_name}.png".replace(' ','_')
    handle_figure(fig, filename)

def plot_feature_contributions_m3(contribs: np.ndarray,
                                  feature_names: list,
                                  graph_id: str):
    """
    Plot a heatmap of feature volume contributions in m³.
    - contribs: N×F array of contributions (node × feature)
    - feature_names: list of length F
    - graph_id: identifier for title/filename
    """
    N, F = contribs.shape
    vmax = np.abs(contribs).max()
    fig, ax = plt.subplots(
        figsize=(F * 0.6 + 1, N * 0.6 + 1), facecolor='white'
    )
    im = ax.imshow(
        contribs,
        cmap='seismic',
        vmin=-vmax,
        vmax=vmax,
        aspect='auto',
        alpha=0.7
    )
    cbar = fig.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
    cbar.set_label('Contribution to Volume (m³)', rotation=270, labelpad=15)
```

I. Appendix
I.2 Code Developed For The Research

```
    ax.set_xticks(np.arange(F))
    ax.set_xticklabels(feature_names, rotation=45, ha='right')
    ax.set_yticks(np.arange(N))
    ax.set_yticklabels([f"Node {i+1}" for i in range(N)])
    ax.set_xlabel('Feature')
    ax.set_ylabel('Node')
    ax.set_title(f'Feature Contributions to Volume (m³) – {graph_id}')

    fig.tight_layout()
    handle_figure(fig, f"heatmap_contrib_m3_{graph_id}.png")

# ==========================
# 8. Main Execution
# ==========================
def main():
    global CURRENT_SESSION_FOLDER
    start_time = time.time()
    logger.info(f"Script start | device={DEVICE} | train={TRAIN}")

    # Prepare viz folder if needed
    if ENABLE_VISUALIZATIONS and not PLOT_VISUALIZATIONS:
        CURRENT_SESSION_FOLDER = get_new_viz_session_folder(VIZ_FOLDER)

    # Load & split in-distribution data
    id_graphs = create_dataset(JSON_FOLDER)
    random.seed(SEED)
    random.shuffle(id_graphs)
    split   = int(len(id_graphs) * TRAIN_SPLIT)
    train_g = id_graphs[:split]
    val_g   = id_graphs[split:]

    # Dataloader for training
    train_loader = None
    if TRAIN:
        train_loader = GraphDataLoader(
            train_g, batch_size=BATCH_SIZE, shuffle=True,
            collate_fn=collate_fn, drop_last=False, num_workers=0
        )

    # Initialize model, loss, optimizer
    model   = GraphRegressor().to(DEVICE)
    loss_fn = nn.HuberLoss()  # Robust to outliers
    optimizer = torch.optim.AdamW(
        model.parameters(),
        lr=LEARNING_RATE,
        weight_decay=1e-4  # Use weight decay for true L2 regularization
    )

    # Ensure save folder exists
    os.makedirs(MODEL_SAVE_FOLDER, exist_ok=True)
    model_path = os.path.join(MODEL_SAVE_FOLDER, "graph_regressor_state.pth")

    # Train or load model
    if TRAIN:
        tl, va, vl, vr = train_and_evaluate(
            model, train_loader, val_g,
            optimizer, loss_fn, DEVICE, NUM_EPOCHS
        )
```



```
        torch.save(model.state_dict(), model_path)
        logger.info(f"Model saved to {model_path}")
        if ENABLE_VISUALIZATIONS:
            plot_training_metrics(tl, va, vl)
    else:
        model.load_state_dict(torch.load(model_path, map_location=DEVICE))
        model.eval()
        logger.info(f"Loaded model from {model_path}")

    # In-distribution evaluation
    id_mape, id_acc, id_r2 = evaluate_model(model, val_g, DEVICE)
    logger.info(f"ID Eval -> MAPE={id_mape:.2f}% Acc={id_acc:.2f}% R2={id_r2:.4f}")
    if ENABLE_VISUALIZATIONS:
        p_id, a_id = gather_predictions(model, val_g, DEVICE)
        for i, name in enumerate(OUTPUT_FEATURE_NAMES):
            plot_prediction_vs_actual_scatter(p_id, a_id, i, name, 'ID')

    # Out-of-distribution evaluation
    ood_g  = create_dataset(EXTRA_VALIDATION_FOLDER)
    ood_gt = create_dataset_dict(EXTRA_GROUND_TRUTH_FOLDER)
    if ood_g:
        ood_mape, ood_acc, ood_r2 = evaluate_extra_validation(
            model, ood_g, ood_gt, DEVICE
        )
        logger.info(f"OOD Eval -> MAPE={ood_mape:.2f}% Acc={ood_acc:.2f}% R2={ood_r2:.4f}")

        if ENABLE_VISUALIZATIONS:
            p_o, a_o = gather_predictions_ood(model, ood_g, ood_gt, DEVICE)
            for i, name in enumerate(OUTPUT_FEATURE_NAMES):
                plot_prediction_vs_actual_scatter(p_o, a_o, i, name, 'OOD')

        # Print per-node predictions for each OOD graph
        for g in ood_g:
            preds = predict(model, g, DEVICE).numpy()
            print(f"\nPredictions for OOD graph {g.design_id}:")
            for node_id, pred in zip(g.node_ids, preds):
                print(f"  Node {node_id}: "
                      f"ObjectCount={pred[0]:.4f}, "
                      f"ObjectVolume={pred[1]:.4f}")

            # --- FEATURE CONTRIBUTION HEATMAPS IN M³ ---
            if ENABLE_VISUALIZATIONS:
                gdev = g.to(DEVICE)
                gdev.ndata['feat'].requires_grad_(True)
                out    = model(gdev, gdev.ndata['feat'])
                vol    = out[:, 1].sum()
                grads  = torch.autograd.grad(vol, gdev.ndata['feat'])[0]  # [N,F]
                inputs = gdev.ndata['feat']
                contribs = (grads * inputs).detach().cpu().numpy()

                plot_feature_contributions_m3(contribs,
                                              INPUT_FEATURE_NAMES,
                                              g.design_id)

    logger.info(f"Done in {time.time() - start_time:.1f}s")

if __name__ == "__main__":
    # Set random seeds for reproducibility
    random.seed(SEED)
    np.random.seed(SEED)
    torch.manual_seed(SEED)
    if torch.cuda.is_available():
        torch.cuda.manual_seed_all(SEED)
    main()
```

I. Appendix
I.2 Code Developed For The Research

I.2.2 Translating The User & Training Design To The JSON Files

```
import Rhino
import rhinoscriptsyntax as rs
import scriptcontext as sc
import math
import re
import os
import glob
import json
from Rhino.Geometry.Intersect import Intersection

# ------------------------------------------------------------------------
# Debugging Helper
# ------------------------------------------------------------------------
def debug_print(msg):
    print("[DEBUG]: {}".format(msg))

# ------------------------------------------------------------------------
# Helper function: Round float values to 3 decimals
# ------------------------------------------------------------------------
def r3(val):
    if isinstance(val, float):
        return round(val, 3)
    elif isinstance(val, list):
        return [r3(x) for x in val]
    else:
        return val

# ------------------------------------------------------------------------
# Discrete Mapping Dictionaries for Box Dimensions
# ------------------------------------------------------------------------
length_dict = {
    1.0 : 918, 1.5 : 1374, 2.0 : 1528, 2.5 : 1984,
    3.0 : 2138, 3.5 : 2594, 4.0 : 2748, 4.5 : 3204,
    5.0 : 3358, 5.5 : 3814, 6.0 : 3968, 6.5 : 4424,
    7.0 : 4578, 7.5 : 5034, 8.0 : 5188, 8.5 : 5644,
    9.0 : 5798, 9.5 : 6254, 10.0: 6408
}

width_dict = {
    1.0 : 918, 1.5 : 1374, 2.0 : 1528, 2.5 : 1984,
    3.0 : 2138, 3.5 : 2594, 4.0 : 2748, 4.5 : 3204,
    5.0 : 3358
}

def map_dimension(measured, mapping_dict):
    best_key = None
    best_diff = None
    for k, v in mapping_dict.items():
        diff = abs(v - measured)
        if best_diff is None or diff < best_diff:
            best_diff = diff
            best_key = k
    return best_key

# ------------------------------------------------------------------------
```



```
# Opening Classification by Curve Length
# ------------------------------------------------------------------------
def classify_opening_by_length(curve, door_target=6706.0, tol=10.0):
    try:
        curve_length = curve.GetLength()
        debug_print("Curve length: {:.3f} mm".format(curve_length))
        return 0 if abs(curve_length - door_target) <= tol else 1
    except Exception as e:
        debug_print("Error computing curve length: {}".format(e))
        return 1

# ------------------------------------------------------------------------
# KOZIJN Mesh Classification Helper
# ------------------------------------------------------------------------
def classify_kozijn_mesh(mesh, door_target=6706.0, tol=10.0):
    bbox = mesh.GetBoundingBox(True)
    dx = bbox.Max.X - bbox.Min.X
    dy = bbox.Max.Y - bbox.Min.Y
    dz = bbox.Max.Z - bbox.Min.Z
    length_measure = max(dx, dy, dz)
    debug_print("KOZIJN mesh classification: max dimension = {:.3f} mm".format(length_measure))
    return 0 if abs(length_measure - door_target) <= tol else 1

# ------------------------------------------------------------------------
# Geometry & Debugging Helpers
# ------------------------------------------------------------------------
def debug_brep_info(brep, index):
    debug_print("Processing Brep index: {}".format(index))
    if not brep:
        debug_print("  -> Brep is None, skipping.")
        return None
    bbox = brep.GetBoundingBox(True)
    cpt  = bbox.Center
    diag = bbox.Diagonal
    length = abs(diag.X)
    width  = abs(diag.Y)
    height = abs(diag.Z)
    min_z = bbox.Min.Z
    debug_print("  -> BoundingBox center: {}".format(cpt))
    debug_print("  -> BBox Dimensions (L x W x H): {:.3f} x {:.3f} x {:.3f}".format(length, width, height))
    debug_print("  -> Minimum Z value: {:.3f}".format(min_z))
    face_areas = []
    for fidx, face in enumerate(brep.Faces):
        fb = face.ToBrep()
        if fb:
            amp_face = Rhino.Geometry.AreaMassProperties.Compute(fb)
            a_val = amp_face.Area if amp_face else 0.0
            face_areas.append(a_val)
            debug_print("    -> Face {} area: {:.3f}".format(fidx, a_val))
        else:
            face_areas.append(0.0)
            debug_print("    -> Face {} => face.ToBrep() failed, set to 0.".format(fidx))
    return {"center": cpt, "dimensions": (length, width, height), "face_areas": face_areas, "min_z": min_z}

def planar_face(face, tolerance):
    srf = face.UnderlyingSurface()
    if not srf or not srf.IsPlanar(tolerance):
        return False, None
    rc, plane = srf.TryGetPlane(tolerance)
    return (rc, plane) if rc else (False, None)

def face_subtract_curve_area(brep_record, face_index, curve_area):
    old_area = brep_record["face_areas"][face_index]
    new_area = max(old_area - curve_area, 0.0)
```

I. Appendix
I.2 Code Developed For The Research

```
    brep_record["face_areas"][face_index] = new_area
    debug_print("    -> Face {}: subtracted {:.3f} (old: {:.3f}, new: {:.3f})".format(
        face_index, curve_area, old_area, new_area))

# ------------------------------------------------------------------------
# BLOCK DEFINITION LOOKUP & OBJECT MESH COUNT
# ------------------------------------------------------------------------
def get_block_definition_by_name(block_name):
    if not block_name:
        return None
    olddoc = sc.doc
    try:
        sc.doc = Rhino.RhinoDoc.ActiveDoc
        bd = sc.doc.InstanceDefinitions.Find(block_name, False)
        debug_print("Block definition lookup for '{}': {}".format(block_name, "Found" if bd else "Not Found"))
        return bd
    finally:
        sc.doc = olddoc

def count_lvl_meshes_and_volume(inst_def):
    if not inst_def: return 0, 0.0, {}
    mesh_count = 0
    total_vol_mm3 = 0.0
    mesh_dict = {}
    seen_ids = set()
    olddoc = sc.doc
    try:
        sc.doc = Rhino.RhinoDoc.ActiveDoc
        for obj in inst_def.GetObjects():
            if obj.ObjectType == Rhino.DocObjects.ObjectType.Mesh \
              and obj.Attributes.IsInstanceDefinitionObject \
              and obj.Id not in seen_ids:
                seen_ids.add(obj.Id)
                name = obj.Attributes.Name or ""
                if name.upper().startswith("LVL"):
                    geo = obj.Geometry
                    if geo.Vertices.Count and geo.Faces.Count:
                        mesh_count += 1
                        mesh_dict[str(obj.Id)] = {"name": name, "geo": geo}
                        vmp = Rhino.Geometry.VolumeMassProperties.Compute(geo)
                        total_vol_mm3 += (vmp.Volume if vmp else 0.0)
    finally:
        sc.doc = olddoc
    debug_print("Counted {} LVL meshes, volume {:.3f} mm^3".format(mesh_count, total_vol_mm3))
    return mesh_count, total_vol_mm3, mesh_dict

def count_kozijn_objects_and_volume(inst_def):
    if not inst_def: return 0, 0.0, {}, 0, 0
    mesh_count = 0
    total_vol_mm3 = 0.0
    mesh_dict = {}
    door_count = 0
    window_count = 0
    seen_ids = set()
    olddoc = sc.doc
    try:
        sc.doc = Rhino.RhinoDoc.ActiveDoc
```



```
        for obj in inst_def.GetObjects():
            if obj.ObjectType == Rhino.DocObjects.ObjectType.Mesh \
              and obj.Attributes.IsInstanceDefinitionObject \
              and obj.Id not in seen_ids:
                seen_ids.add(obj.Id)
                name = obj.Attributes.Name or ""
                if name.upper().startswith("KOZIJN"):
                    geo = obj.Geometry
                    if geo.Vertices.Count and geo.Faces.Count:
                        mesh_count += 1
                        mesh_dict[str(obj.Id)] = {"name": name, "geo": geo}
                        cls = classify_kozijn_mesh(geo)
                        if cls == 0:
                            door_count += 1
                        else:
                            window_count += 1
                        vmp = Rhino.Geometry.VolumeMassProperties.Compute(geo)
                        total_vol_mm3 += (vmp.Volume if vmp else 0.0)
    finally:
        sc.doc = olddoc
    debug_print("Counted {} KOZIJN meshes: {} doors, {} windows, vol {:.3f} mm^3".format(
        mesh_count, door_count, window_count, total_vol_mm3))
    return mesh_count, total_vol_mm3, mesh_dict, door_count, window_count

# ------------------------------------------------------------------------
# Functions for Handling Multiple Design Iterations
# ------------------------------------------------------------------------
def get_bounding_box(rec):
    center = rec["center"]
    dims = rec["dimensions"]
    half = [d / 2 for d in dims]
    return ((center.X-half[0], center.Y-half[1], center.Z-half[2]),
            (center.X+half[0], center.Y+half[1], center.Z+half[2]))

def bbox_intersect(b1, b2, threshold=0.0):
    for i in range(3):
        if b1[1][i] + threshold < b2[0][i] or b2[1][i] + threshold < b1[0][i]:
            return False
    return True

def build_adjacency_graph(breps_info, threshold=0.0):
    n = len(breps_info)
    adj = {i: [] for i in range(n)}
    for i in range(n):
        if not breps_info[i]: continue
        b1 = get_bounding_box(breps_info[i])
        for j in range(i+1, n):
            if not breps_info[j]: continue
            b2 = get_bounding_box(breps_info[j])
            if bbox_intersect(b1, b2, threshold):
                adj[i].append(j)
                adj[j].append(i)
                debug_print(f"Brep {i} and {j} are connected.")
    return adj

def find_connected_components(adj, n):
    visited = [False]*n
    comps = []
    def dfs(u, comp):
        visited[u] = True
        comp.append(u)
        for v in adj[u]:
            if not visited[v]:
                dfs(v, comp)
```

I. Appendix
I.2 Code Developed For The Research

```
    for i in range(n):
        if not visited[i]:
            comp = []
            dfs(i, comp)
            comps.append(comp)
            debug_print(f"Found component: {comp}")
    return comps

# ------------------------------------------------------------------------
# MAIN GHPYTHON SCRIPT
#
# Inputs: x (Breps), y (Curves), z (block name strings), json_path (string), delete_flag (bool)
# ------------------------------------------------------------------------
debug_print("=== Starting Brep records processing ===")
breps_info = []
brep_data = []

# 1) Process each Brep from input 'x'
for i, brep in enumerate(x):
    debug_print(f"Processing Brep {i} in main loop.")
    info = debug_brep_info(brep, i)
    if not info:
        breps_info.append(None)
        brep_data.append("None")
        continue
    rec = {
        "brep_index": i,
        **info,
        "block_name": None,
        "lvl_mesh_cnt": 0,
        "lvl_mesh_volume_mm3": 0.0,
        "lvl_mesh_dict": {},
        "kozijn_mesh_cnt": 0,
        "kozijn_mesh_volume_mm3": 0.0,
        "kozijn_mesh_dict": {},
        "openings": {"count":0,"total_area":0.0,"door_count":0,"window_count":0}
    }
    breps_info.append(rec)
    debug_print(f"Brep record created for index {i}.")

# 2) Tolerance from document
tolerance = sc.doc.ModelAbsoluteTolerance
debug_print(f"Using model absolute tolerance: {tolerance:.6f}")

# 3) Robust Curve Processing
debug_print("=== Starting robust curve processing ===")
if y:
    for j, crv in enumerate(y):
        debug_print(f"Processing curve index {j}.")
        if not crv:
            debug_print(f"Curve {j} is None; skipping.")
            continue
        amp = Rhino.Geometry.AreaMassProperties.Compute(crv)
        if not amp:
            debug_print(f"Curve {j} invalid; skipping.")
            continue
        opening_area = amp.Area
        matched = False
```



```
        for bi, rec in enumerate(breps_info):
            if not rec: continue
            if not x[bi].GetBoundingBox(True).Contains(amp.Centroid):
                continue
            cls = classify_opening_by_length(crv, tol=tolerance)
            rec["openings"]["count"]      += 1
            rec["openings"]["total_area"] += opening_area
            if cls == 0:
                rec["openings"]["door_count"] += 1
            else:
                rec["openings"]["window_count"] += 1
            matched = True
            break
        if not matched:
            debug_print(f"Curve {j} did not match any Brep.")
else:
    debug_print("No curves to process; skipping.")

# 4) Block name matching
debug_print("=== Starting block name matching ===")
for i, rec in enumerate(breps_info):
    if not rec: continue
    if i < len(z):
        mm = re.search(r"\(([^()]+)\)", z[i])
        rec["block_name"] = mm.group(1) if mm else z[i]
        debug_print(f"Brep {i} block name: {rec['block_name']}")
    else:
        debug_print(f"Brep {i} => no matching block name.")

# 5) Count LVL and KOZIJN meshes & volumes
debug_print("=== Starting mesh count and volume aggregation ===")
for i, rec in enumerate(breps_info):
    if not rec or not rec["block_name"]: continue
    inst_def = get_block_definition_by_name(rec["block_name"])
    if not inst_def:
        debug_print(f"No block definition for '{rec['block_name']}'.")
        continue
    mc, vol_mm3, md = count_lvl_meshes_and_volume(inst_def)
    rec["lvl_mesh_cnt"] = mc
    rec["lvl_mesh_volume_mm3"] = vol_mm3
    rec["lvl_mesh_dict"] = md
    mc2, vol2, md2, dcnt, wcnt = count_kozijn_objects_and_volume(inst_def)
    rec["kozijn_mesh_cnt"] = mc2
    rec["kozijn_mesh_volume_mm3"] = vol2
    rec["kozijn_mesh_dict"] = md2

# 6) Build adjacency graph and components
debug_print("=== Building adjacency graph ===")
adjacency = build_adjacency_graph(breps_info, threshold=1.0)
components = find_connected_components(adjacency, len(breps_info))
debug_print(f"Found {len(components)} design iterations: {components}")

# 7) Compute SideIntersectionArea using Intersection.BrepBrep
side_intersection_areas = {i: 0.0 for i in range(len(breps_info))}
for comp in components:
    for idx1 in range(len(comp)):
        i = comp[idx1]
        rec1 = breps_info[i]
        if not rec1: continue
        for idx2 in range(idx1+1, len(comp)):
            j = comp[idx2]
            rec2 = breps_info[j]
            if not rec2: continue
            res, curves, pts = Intersection.BrepBrep(x[i], x[j], tolerance)
```

I. Appendix
I.2 Code Developed For The Research

```
        if res and curves:
            joined = Rhino.Geometry.Curve.JoinCurves(curves, tolerance)
            for crv in joined:
                if crv and crv.IsClosed:
                    amp = Rhino.Geometry.AreaMassProperties.Compute(crv)
                    if amp:
                        a_mm2 = amp.Area
                        side_intersection_areas[i] += a_mm2
                        side_intersection_areas[j] += a_mm2
                        debug_print(f"Intersection Brep {i}-{j}: {a_mm2:.3f} mm^2")

# 8) JSON cleanup & numbering
existing = glob.glob(os.path.join(json_path, "design_*.json"))
if delete_flag:
    debug_print("delete_flag=True -> removing existing JSONs")
    for f in existing:
        try:
            os.remove(f)
        except Exception as e:
            debug_print(f"Failed to remove {f}: {e}")
    start_num = 1
else:
    debug_print("delete_flag=False -> preserving existing JSONs")
    nums = []
    for f in existing:
        m = re.match(r".*design_(\d+)\.json$", os.path.basename(f))
        if m:
            nums.append(int(m.group(1)))
    start_num = (max(nums) + 1) if nums else 1
debug_print(f"Starting design number: {start_num}")

# 9) Process each design iteration and write JSON
design_counter = start_num - 1
for comp in components:
    design_counter += 1
    did = f"design_{design_counter:03d}"
    debug_print(f"Processing {did} with Brep indices {comp}")

    # Build connectivity lookup
    conn = {}
    for i in comp:
        nid = f"{did}_brep_{breps_info[i]['brep_index']:04d}"
        conn[nid] = [
            f"{did}_brep_{breps_info[j]['brep_index']:04d}"
            for j in adjacency[i] if j in comp
        ]

    # Build node entries
    nodes = []
    for i in comp:
        rec = breps_info[i]
        if not rec: continue
        dims = rec["dimensions"]
        length_key = map_dimension(dims[0], length_dict)
        width_key  = map_dimension(dims[1], width_dict)
        level_val  = 1 if rec["min_z"] > 0 else 0
```



```
        total_face   = sum(rec["face_areas"])
        area_after   = total_face - rec["openings"]["total_area"]
        obj_count    = rec["lvl_mesh_cnt"] + rec["kozijn_mesh_cnt"]
        obj_vol      = rec["lvl_mesh_volume_mm3"] + rec["kozijn_mesh_volume_mm3"]
        side_m2      = side_intersection_areas[i] / 1e6

        nid = f"{did}_brep_{rec['brep_index']:04d}"
        entry = {
            "NodeID": nid,
            "length_m":      r3(dims[0] / 1000),
            "width_m":       r3(dims[1] / 1000),
            "lengthkey":     r3(length_key),
            "widthkey":      r3(width_key),
            "DoorCount":     rec["openings"]["door_count"],
            "WindowCount":   rec["openings"]["window_count"],
            "OpeningArea":   r3(rec["openings"]["total_area"] / 1e6),
            "ObjectCount":   obj_count,
            "ObjectVolume":  r3(obj_vol / 1e9),
            "AreaAfterOpenings": r3(area_after / 1e6),
            "SideIntersectionArea": r3(side_m2),
            "connectivity":  conn.get(nid, []),
            "Level":         level_val
        }
        nodes.append(entry)

    # Add total_modules
    total_modules = len(nodes)
    for node in nodes:
        node["total_modules"] = total_modules

    # Write out JSON
    design_json = {"design_id": did, "nodes": nodes}
    out_path = os.path.join(json_path, f"{did}.json")
    try:
        with open(out_path, "w") as f:
            json.dump(design_json, f, indent=4)
        debug_print(f"Wrote JSON: {out_path}")
    except Exception as e:
        debug_print(f"Failed to write {out_path}: {e}")

debug_print("Script complete.")
```

I. Appendix
I.3 Research Trajectory

P1

Following the initial presentation, the proposed direction was deemed sound; however, several key issues require definition before proceeding. Research into established methods and algorithms was then initiated, and both the methodology and overall research framework were defined.

A data requirements sheet and corresponding calculation methods were proposed to enable process monitoring and evaluation.

P2

Dataset development commenced immediately. After the second presentation, the project's significance and novelty were recognized, but more precise, high-level definitions to ensure that non-specialists could fully grasp the research objectives, were requested.

After 2 months, an internship agreement was formalized with Sustainer, granting access to its toolkit—the primary platform for generating the 3D models used to construct the training dataset.

Sustainer contributed to the project's research and development through weekly consultations and by clarifying the relationship between carbon metrics and designer inputs.

Challenges emerged over the number of samples produced with the Sustainer toolkit, so the company requested a clear target and generation strategy before moving forward. Generating detailed building modules proved computationally intensive and very time consuming as module counts increased, and achieving sufficient prediction accuracy required producing a large volume of design samples.

Latin Hypercube Sampling was explored to guide dataset creation, since uncontrolled output scenarios could number in the hundreds of thousands,or even millions,making a focused subset essential.

An ML data schema was established, defining input and target features in collaboration with the project supervisors and Sustainer, and a validation protocol was developed to assess GNN performance and data quality.

P3

Both supervisors emphasized the need to explain why a Graph Neural Network was chosen and to develop two visual frameworks: one that converts a design into a graph for GNN processing and performance evaluation, and another that translates the model's outputs into clear, designer- and stakeholder-friendly feedback.

Early tests revealed rapid, misleading accuracy gains that collapsed on unseen designs. Although optimization improved training, feature-attribution checks showed the model hadn't learned the underlying physics. Consequently, alternative architectures were evaluated.

P4

After experimenting with various optimizers, loss functions, GNN architectures, dropout rates and activation functions (all detailed in the report), the model exhibited healthy training with steadily improving accuracy and no overfitting. However, once accuracy reached 85%-90%, further training caused the model to underfit.

After selecting the GNN architecture that demonstrated sound feature attribution and stable training, early stopping was implemented to prevent underfitting and preserve the model's ability to capture the underlying physical relationships.

Once the results were deemed satisfactory, the GNN script was refactored into modular components, enabling feature extensions and the integration of multiple visualization methods without impacting core functionality.

With satisfactory results achieved, WP4 began with the development of a Grasshopper plugin. The goal was to build a prediction-only version of the GNN and Grasshopper components that generate Rhino visualizations of its outputs for end users.

The main challenge was delivering feature attributions without retraining the model for each new design. This capability was implemented but remains optional due to its high computational cost.

Once development produced satisfactory results, visualizations were refined, feedback was gathered, and a strategy for future enhancements was formulated.

P5

Between P4 and P5, feedback from the formal assessment will be incorporated and visualizations refined to enhance the research deliverables and support future development and scalability.

After achieving baseline functionality, the GNN was tested on increasingly complex design scenarios to identify limitations and weaknesses for future improvement.

Figure 89: Computational research trajectory / Source: Author, 2025.

I. Appendix
I.3 Research Trajectory

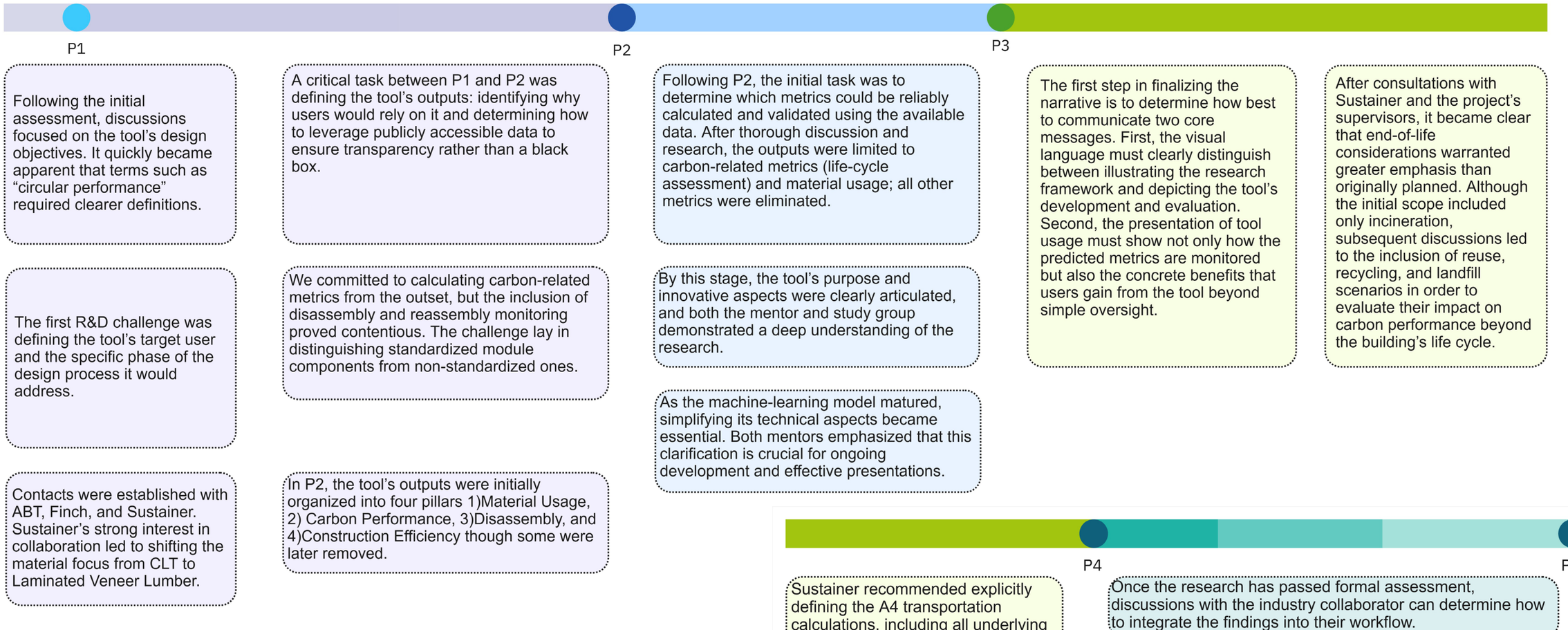


Figure 90: Circularity research trajectory / Source: Author, 2025.

BIBLIOGRAPHY & APPENDIX

I. Appendix
I.4 Data Requirements And Specifications Sheet

The tables that follow define the exact data inputs, formats, and structures required to drive both the LCA and GNN components of our LVL-based prototype, although to the full background infromation on the reasoning and argumentation behind the selection of the following data, is located in the next chapters, this section here works only as a straightforward and contained display of infromation.

Environmental Product Declaration Data

Variable	Description	Unit
1) Product Composition		
Material	Name of material component	text
Weight	Mass of material per functional unit	kg
2) Packaging		
Packaging material	Type of packaging component	text
Packaging weight	Mass of packaging per functional unit	kg
3) Transportation		
Bulk density	Bulk density of product	$\mathrm{kg\,m^{-3}}$
Vehicle fuel consumption (truck)	Fuel use per t · km by truck	$L/(\mathrm{t\cdot km})$
Vehicle fuel consumption (ferry)	Fuel use per t · km by ferry	$L/(\mathrm{t\cdot km})$
Emission factor (truck)	CO_2 per t · km by truck	$\mathrm{kg\,CO_2/(t\cdot km)}$
Emission factor (ferry)	CO_2 per t · km by ferry	$\mathrm{kg\,CO_2/(t\cdot km)}$
Coordinates	Production location (lat, long)	text
4) Mandatory Impact Category Indicators (EN 15804+A2)		
GWP total	Total global warming potential (A1–D)	$\mathrm{kg\,CO_2/m^3}$
GWP-biogenic	Biogenic carbon uptake or emissions	$\mathrm{kg\,CO_2/m^3}$
PERT	Primary energy (renewable + non-renewable)	MJ
PENRT	Non-renewable primary energy total	MJ
5) End-of-Life (EoL) Scenarios		
Incineration mandatory; Reuse, Recycling, Landfill optional		

Related work packages : WP2,WP4

Translating Designs to GNN-Training Samples (JSON)

Variable	Description	Notes
Inputs		
buildingModule	Brep geometries to process	Rhino Brep objects
openingPlaceholder	Curves defining openings	Rhino Curve objects
sustainerBlockDefinition	Instance definition names	Block name labels
json_path	Folder for JSON output	Filesystem directory
delete_flag	Clear previous JSONs	yes / no
Parameters & Mappings		
length_dict	Maps nominal size → millimetres	Python dict
width_dict	Maps nominal width → millimetres	Python dict
door_target	Threshold length to classify doors	mm
tol	Geometry tolerance	mm

JSON Data Structure

Field	Description	Notes
design_id	Design iteration identifier	integer
NodeID	Module node identifier	integer
length_m	Module length (m, 3-decimal)	m
width_m	Module width (m, 3-decimal)	m
lengthkey	Nominal length key	nominal size
widthkey	Nominal width key	nominal size
DoorCount	Number of door openings	integer
WindowCount	Number of window openings	integer
OpeningArea	Total opening area (m^2, 3-decimal)	m^2
ObjectCount	Number of mesh objects	integer
ObjectVolume	Module volume (m^3, 3-decimal)	m^3
AreaAfterOpenings	Surface area post-openings (m^2)	m^2
SideIntersectionArea	Intersection area (m^2, 3-decimal)	m^2
connectivity	Adjacent module node IDs	integer list
Level	Vertical level (0 = ground, 1 = above)	integer
total_modules	Total modules in iteration	integer

Related work packages : WP3,WP4

I. Appendix
I.4 Data Requirements And Specifications Sheet

Echo Plugin – Required Data Variables

Project Setup		
Inputs		
Manufacturer	Chosen product manufacturer	e.g. Stora Enso
LCA Scenario	Selected life-cycle stages	e.g. Cradle to Grave
Project Location	Address for haulage calculation	Full postal address
End-Of-Life Scenario	Chosen end-of-life treatment	e.g. Landfill
Outputs		
EPD ID	EPD record identifier	integer
EPD_info	Summary of EPD carbon results	descriptive text
LCA ID	LCA scenario identifier	integer
LCA_info	Summary of LCA energy results	descriptive text
Project Distance	Haulage distances	[truck_km, ferry_km]
EOL ID	End-of-life scenario identifier	integer
EOL_info	Summary of EoL parameters	descriptive text
Material Calculator		
Note: uses EPD ID, LCA ID & Project Distance as additional inputs.		
Inputs		
modules	Module geometries	Brep objects
openings	Opening curves	Curve objects
toggle	Enable detailed breakdown	yes / no
Outputs		
Volume	Predicted material volume	m^3
Weight	Predicted material weight	kg
Transportation Emissions	Haulage CO_2 emissions	descriptive text
Feature Contributions	Breakdown of contributions	value list
Features: DoorCount, WindowCount, OpeningArea, ObjectCount, ObjectVolume, SideIntersectionArea; each contribution shown in m^3.		

Related work packages : WP4,WP5

BIBLIOGRAPHY & APPENDIX