ECOLOPES

ECOlogical building enveLOPES: a game-changing design approach for regenerative urban ecosystems

H2020-FET-OPEN-2021-2025

Action number 964414

D5.2

ECOLOPES Voxel Model Demo

Dissemination level:	Public
Contractual date of delivery:	Month 30, 30 September 2023
Actual date of delivery:	28 September 2023
Work package:	WP5
Task:	T5.1
Type:	DEMO
Approval Status:	Submitted
Version:	v0.1
Number of pages:	30
Filename:	D5.2_Ecolopes_ECOLOPESVoxelModel_20230930_v0.1.pdf

Abstract

This delivery summarily introduces conceptual and technical characteristics of the ECOLOPES Voxel Model (the detailed description can be found in *D5.3 ECOLOPES Voxel Model*) and focuses on the ECOLOPES Voxel Model demo. The demo describes the steps and tools involved in utilizing the ECOLOPES Voxel Model, especially in connection with the EIM Ontology and the ECOLOPES Computational Model, and the tools enabling the designer to exchange data with the Rhino / Grasshopper environment. This deliverable relates closely to deliverables *D5.3 ECOLOPES Voxel Model*, as well as *D4.2 Interim EIM Ontology* and *D5.4 ECOLOPES Computational Model*.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

co-funded by the European Union

HISTORY

Version	Date	Reason	Revised by
v0.2	27.09.2023	Revised Deliverable	Michael Hensel
	27.09.2023	Revised Deliverable	Jakub Tyc

AUTHOR LIST

Organization	Name	Contact Information
TU Wien	Michael Hensel	michael.hensel@tuwien.ac.at
TU Wien	Jakub Tyc	jakub.tyc@tuwien.ac.at

EXECUTIVE SUMMARY

This delivery contains a demo of the ECOLOPES Voxel Model. We describe the steps and tools involved in utilizing the ECOLOPES Voxel Model (*D5.3 ECOLOPES Voxel Model*) in connection with the other two key components of the ontology-aided generative computational design process: the EIM Ontology (*D4.2 Interim EIM Ontology*) and the ECOLOPES Computational Model (*D5.4 ECOLOPES Computational Model*). This includes in particular the tools that enable the designer to exchange data with the Rhino / Grasshopper environment.

The ECOLOPES Voxel Model consists of a SQL database. Only data that can be expressed spatially is contained within the voxel model. This data is derived from (1) open and expert databases, (2) the ecological model (*D4.1 Preliminary EIM Ontology*), (3) the knowledge generation framework (*D3.2 Draft ECOLOPES platform architecture*), (4) the EIM Ontologies (*D4.2 Interim EIM Ontologies*), and (5) computational simulations, generated in preparation of or as part of the generative phase of the process.

Section 1 summarily introduces conceptual and technical characteristics of the ECOLOPES Voxel Model. The detailed description can be found in *D5.3 ECOLOPES Voxel Model*.

Section 2 contains the ECOLOPES Voxel Model demo.

ABBREVIATIONS AND ACRONYMS

ASP	Answer Set Programming
CAD	Computer Aided Design
DEM	Digital Elevation Model
DSM	Digital Surface Model
EA	Evolutionary Algorithm
EIM	ECOLOPES Information Model
EM	ECOLOPES Ecological Model
GA	Genetic algorithm
GCD	Generative Computational Design
GH	Grasshopper
JSON	JavaScript Object Notation
KGF	Knowledge Generation Framework
KPI	Key Performance Indicator
OWL	Web Ontology Language
RDB	Relational Database
RDF	Resource Description Framework
SQL	Structured Query Language
WP	Work-package
PFGs	Plant Functional Groups
AFGs	Animal Functional Groups

TABLE OF CONTENTS

History **2**

Author list 2

Executive Summary 3

1 Introduction 6

1.1 The ECOLOPES Computational Framework 6

1.2 The Ontology-aided Generative Computational Design Process 7

1.3 The ECOLOPES Voxel Model 7

2 ECOLOPES Voxel Model Demo 15

2.1 Structure of the ECOLOPES Voxel Model 15

2.2 Structuring Spatial Data for Inclusion in the ECOLOPES Voxel Model 16

2.3 Loading data into the ECOLPES Voxel Model 18

2.4 Default Web-base Interface for ECOLOPES Voxel Model Data (pgAdmin) 19

2.5 Rhinoceros / Grasshopper Interface for the ECOLOPES Voxel Model 20

2.5.1 Listing available voxel levels in the ECOLOPES Voxel Model 20

2.5.2 Retrieving metadata from the ECOLOPES Voxel Model 21

2.5.3 Retrieving the voxel index data (vox_idx) and geometric data from the ECOLOPES Voxel Model 21

2.5.4 Retrieving color data from the ECOLOPES Voxel Model 22

2.5.5 Retrieving the list of available data layers from the ECOLOPES Voxel Model 23

2.5.6 Retrieving a single data layer from the ECOLOPES Voxel Model 24

2.6 Inserting new Geometry into the ECOLOPES Voxel Model 24

2.6.1 Simplified method for exporting Rhinoceros 3D geometry to the ECOLOPES Voxel Model 25

2.7 Exporting Data from the ECOLOPES Voxel Model 25

2.7.1 Exporting simplified 2.5D geometry from the ECOLOPES Voxel Model 25

2.8 Integrating the ECOLOPES Voxel Model with the EIM Ontology 27

2.8.1 Querying the ontological representation of the ECOLOPES Voxel Model data in Grasshopper 29

3 Publication Plan 30

References 30

1 Introduction

In section 1 we summarily describe the ECOLOPES Computational Framework to indicate the context and location of the ontology-aided generative computational design process. Secondly, we summarize the ontology-aided generative computational design process which includes three key components: (1) the EIM Ontology (*D4.2 Interim EIM Ontology*), (2) the ECOLOPES Voxel Model, and (3) the ECOLOPES Computational Model (*D5.4 ECOLOPES Computational Model*). Finally, we summarily describe key conceptual and technical characteristics of the ECOLOPES Voxel Model. The detailed description can be found in *D5.3 ECOLOPES Voxel Model*.

1.1 The ECOLOPES Computational Framework

The ECOLOPES computational framework, its technical components and data flow between the latter (computational workflow) was elaborated in WP3 and the updated version presented in D3.3. (M29) (Fig. 1).

The ECOLOPES Computational Framework facilitates informed multi-species design for ecological building envelopes, that we term *ecolopes* (Fig. 1) (Weisser et al. 2022). It includes technical components such as the Ecological Model, the Knowledge Base, the design generation environment, which we term "*ontology-aided generative computational design process*", the Optimization environment, and components for validation. The design generation environment (ontology-aided generative computational design process) developed in WP5 (*D5.3 ECOLOPES Voxel Mode*l and *D5.4 ECOLOPES Computational Model*) facilitates design generation and the generation of design search space populated with alternative solutions that can be analyzed, evaluated and ranked. The ECOLOPES Computational Model provides input for optimization, the output of which provides the basis for the overall validation (WP7) of the ECOLOPES Computational Framework.

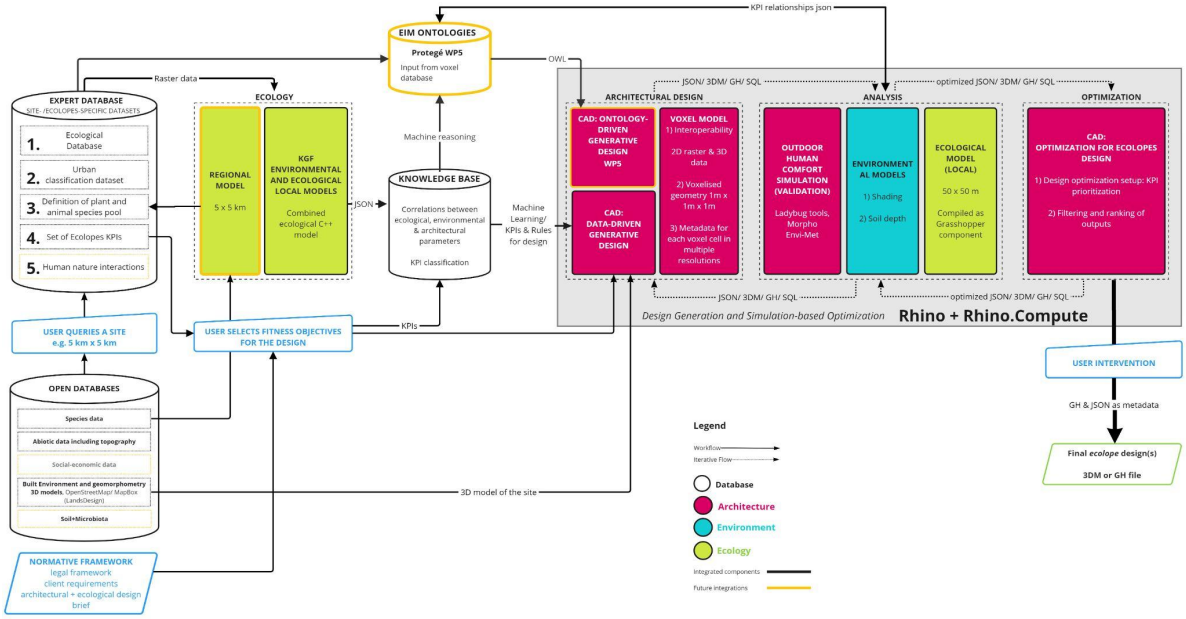

Fig. 1: Ecolopes computational framework showing integrated (black frame) and non-integrated technical components (yellow frame) (D3.3).

1.2 The Ontology-aided Generative Computational Design Process

The ontology-aided generative computational design process consists of three key components: (1) the EIM Ontologies (*D4.2 Interim EIM Ontology*) that guide the design process in its different stage and can be queried by the designer, (2) a voxel model that integrates relevant datasets for the design process (*D5.3 ECOLOPES Voxel Model*), and (3) the ECOLOPES Computational Model (*D5.4 ECOLOPES Computational Model*) which comprises specific algorithmic processes implemented the Rhino / Grasshopper environment.

In the generative computational design process data is collected from different sources, including databases, the Ecological Model, KGF, simulations, etc. This data is utilized with the purpose to generate design variations for a context-specific ecological building envelope. The ECOLOPES Voxel Model offers a key utility for the different stages of the generative computational design process, which includes the *translational* process and (2) the *generative* process. The *translational* process serves to lay out the project-specific problem space for design. In this process requirements of the design brief for a given project and site and additional requirements are analyzed, correlated, spatialized, and prepared for design generation. This process is ontology-aided and involves the preparation of the datasets referred to as *maps* and *networks* (*D5.1 Development Process for ECOLOPES Algorithms*). The *generative* process serves to extend the solution space for design. This entails generation of a range of design outputs that can be evaluated and ranked. This iterative process is ontology-aided and culminates in the generation of (1) spatial organization expressed as the dataset *volumes*, and (2) site and building geometry expressed as the dataset *landform* (*D5.1 Development Process for ECOLOPES Algorithms*). Each design outcome will consist of (1) a CAD model, (2) corresponding datasets contained in the voxel model, and (3) ontological output.

1.3 The ECOLOPES Voxel Model

The ECOLOPES Voxel Model is described in detail in *D5.3 ECOLOPES Voxel Model*. Here we only provide a short summary. To store 3D voxel data, we utilize Relational Databases (RDBs, such as SQL databases) as technology agnostic solution. The ECOLOPES Voxel Model receives data from different sources (databases, EM, KGF, GIS simulations), and the data can be indicated and / or called via the EIM Ontologies. Data contained in the voxel model can then be utilized in the data-driven generative computational design process. The ECOLOPES Voxel Model uses a range of technologies to link the voxel data encoded in an RDB-based voxel model with the Rhinoceros / Grasshopper interface (see *D5.3.*).

In the interdisciplinary ECOLOPES research project researchers have different disciplinary expertise and approaches, as well as varied computational proficiency. For this reason, a self-explanatory and technology agnostic solution was chosen for the ECOLOPES Voxel Model. We utilize Relational Databases (RDBs, such as SQL databases) that are generally widely implemented and most contemporary programming languages contain well developed bindings for RDBs. User interaction will be realized through the ECOLOPES front-end tools (WP3) based on Rhinoceros / Grasshopper; 3D CAD software widely used in the context of architectural design. In the ECOLOPES research project it is necessary to correlate and spatialize data pertaining to different disciplines, including ecological, environmental, and architectural data, as well as data pertaining to landscape architecture. The ECOLOPES Voxel Model receives data from different sources, including relevant databases, the ecological model, the knowledge generation framework, and in project specific cases also various

simulations executed in Geographic Information Systems (GIS) software. Relevant data can be indicated and / or called via the EIM Ontologies. Data contained in the voxel model can then be utilized in the data-driven generative computational design process through which design outputs are created that consist of (1) geometry contained within the CAD model, (2) design specific data contained in the voxel model, and (3) ontological output. The resulting data package can then be used within the optimization process to derive design outputs with optimized architectural and ecological performances. The data integration approach currently implemented in Tasks T5.1, T5.2 and T4.7 supports the ontology-aided generative computational design process for designing ecological building envelopes. A conceptual overview of this data integration approach is shown in Figure 2.

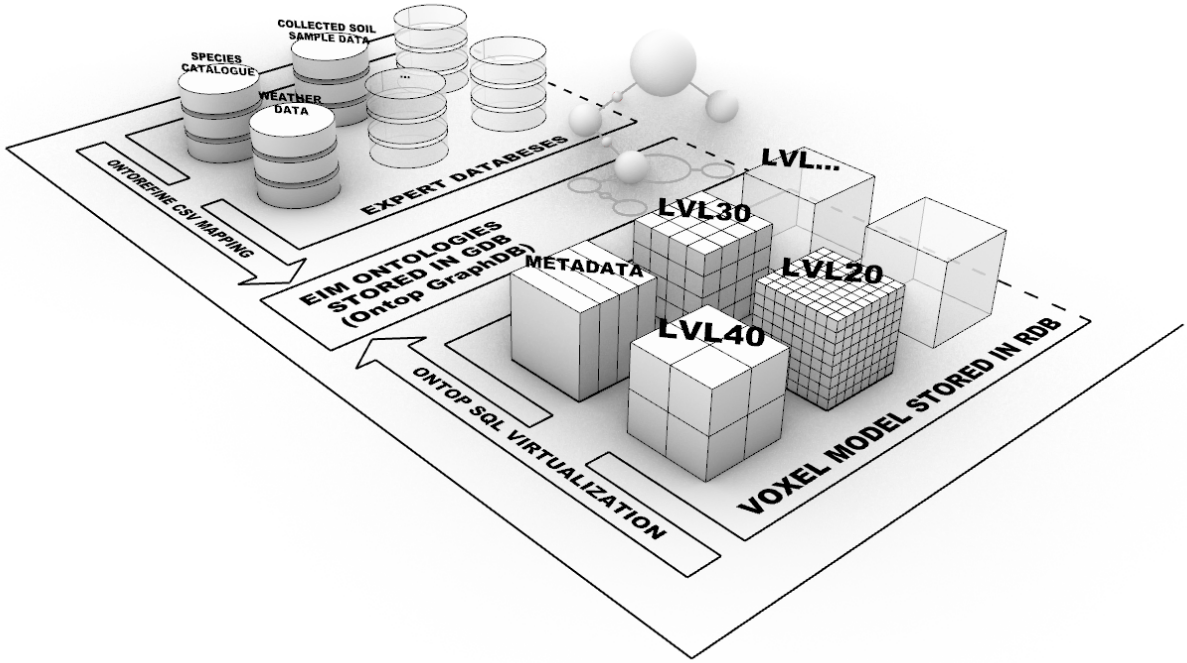

Fig. 2: This updated version of Figure 20 from Deliverable D5.1 illustrates the conceptual approach for systematic data integration and structuring. This approach is being implemented to support the ontology-aided design process. The GraphDB component stores the EIM Ontologies and establishes interfaces with external datasets and the Ecolopes Voxel Model by utilizing Ontop GraphDB virtualization and mapping techniques.

The ECOLOPES Voxel Model stores multi-scalar data that describes geometry and chosen environmental conditions. The EIM Ontology implemented in the GraphDB environment integrates external datasets, such as species occurrence data with spatial data contained in the RDB-based voxel model. The GCD process is initiated by user interaction translated into the Dataset Networks and the computational components are providing feedback based on the GraphDB reasoning capacities. In the ECOLOPES Voxel Model openly available datasets are used to generate the initial datasets contained in the voxel model. To enable integration of the diverse disciplinary datasets, the concept of multi-scalar data was introduced by way of inclusion of *levels* within the voxel model structure (*D5.1 Development Process for ECOLOPES Algorithm*). Each level is introduced into the ECOLOPES Voxel Model to facilitate interaction with an external computational procedure.

The ECOLOPES Voxel Model uses a range of technologies to link the voxel data encoded in an RDB-based voxel model with the Rhinoceros / Grasshopper interface. Figure 3 shows the chosen software technologies that are used in the ECOLOPES Voxel Model implementation.

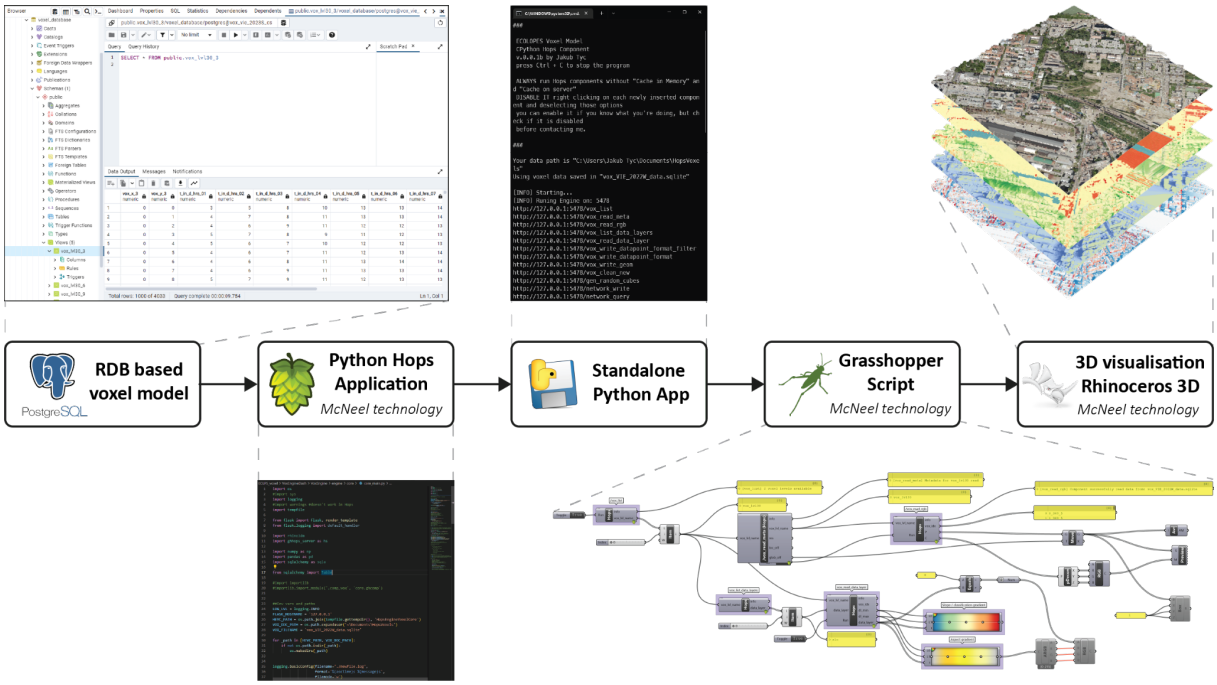

Fig. 3: Technologies utilized to implement the ECOLOPES Voxel Model were linked in a sequence. RDB-based voxel data can be queried through the McNeel Python Hops application packaged into a single executable file. This Python application exposes voxel data in the McNeel Rhinoceros/ Grasshopper environment for user interaction.

This implementation builds on the technologies readily available within the Rhinoceros software ecosystem. Rhinoceros and Grasshopper are widely used tools in architectural design. Originally, McNeel introduced GHPython components into the Grasshopper environment, based on the IronPython (*IronPython*, 2017). To overcome some of the limitations posed by the IronPython, we used the GH Hops components, which adds external functions to GH through Rhino.Compute. Hops integrates a modern Python interpreter (CPython 3.9) with the Rhinoceros / Grasshopper environment through a REST AP-based interface. ECOLOPES Voxel Model GH definitions are written as Hops components to establish an interface with the RDB. The SQLAlchemy Python library (SQLAlchemy, 2018/2023) is used to provide an SQL-dialect-agnostic solution for integrating RDBs with the digital design process implemented in the Rhinoceros software. For the RDB-based ECOLOPES Voxel Model, different types of RDBs, including SQLite, MariaDB and PostgreSQL, have been prototyped and tested. Python technology was chosen due to its wide compatibility. Python version 3.9 is compatible with the McNeel libraries. The Python Hops application has been packaged into a single executable file for internal distribution. The presented application has been successfully tested on both Windows and MacOS platforms, including the ARM based M1 architecture. Data contained in the ECOLOPES Voxel Model has been created with a range of open-source geospatial analysis tools, such as QGIS (Open-Source Geospatial Foundation Project 2020), Whitebox Tools (Lindsay 2016) and SAGA GIS (Conrad et al. 2015). Figure 4 shows the datasets that are currently encoded in the ECOLOPES Voxel Model.

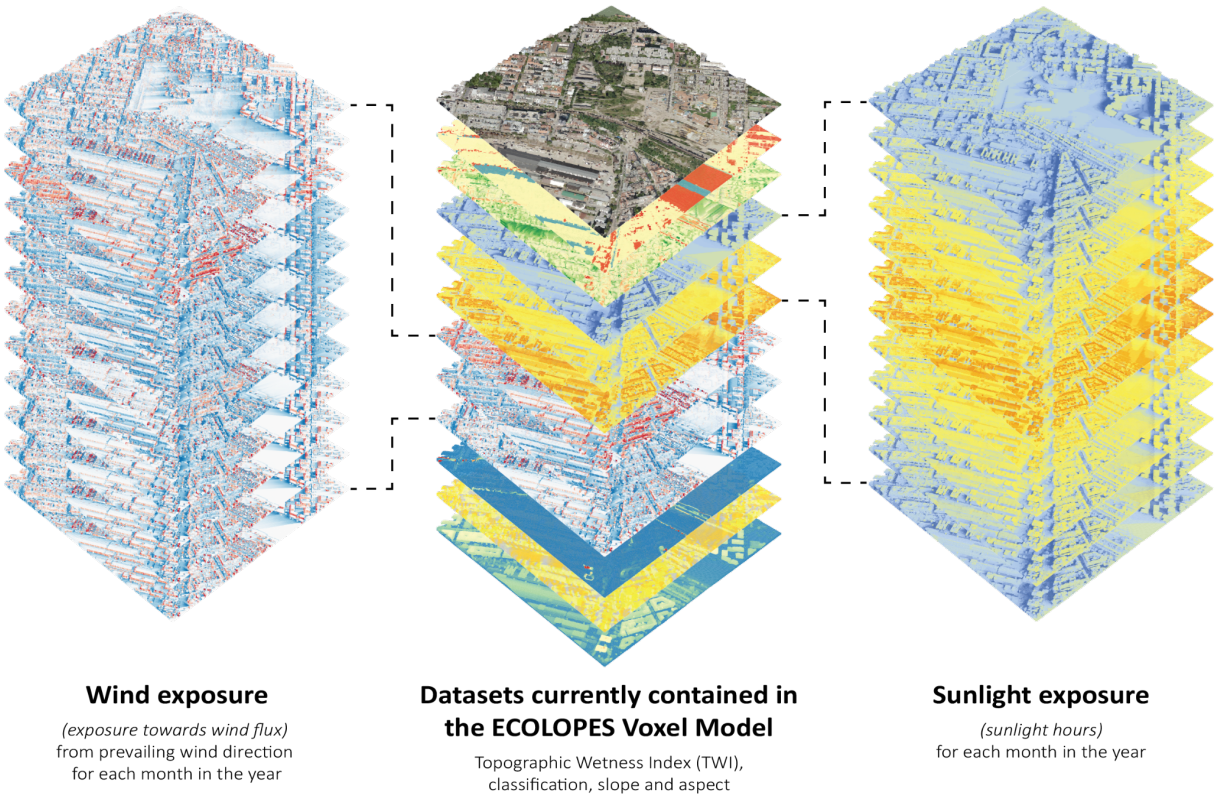

Fig. 4: Datasets contained in the ECOLOPES Voxel Model include geometric and classification data. This includes environmental performance data such as, for instance, topographic wetness index, as well as time series data describing insolation time and wind exposure.

The selected datasets are representative of different aspects of environmental performance that are often used in urban planning (Wilson & Gallant, 2000; Conrad et al., 2015). The datasets include solar and wind exposure, as well as topographic water-related conditions. For solar exposure the average insolation time for each month in a year was computed. This parameter, often described as sunlight hours, is commonly used to evaluate sunlight availability in urban contexts. Topographic wetness conditions were evaluated using the Topographic Wetness Index (TWI).

The role of levels within the structure of the voxel model facilitates specific functionality in the GCD process. On the technological level, data expressed as levels in the Voxel Model is materialized in the PostgreSQL environment either as SQL tables or as SQL table views. This enables multiple external components to interact with the RDB-based voxel model by querying the voxel data from the chosen level. The requirement of the GCD processes to operate in different scales necessitates changes of extent and resolution to be handled within the RDB environment. The architecture of the SQL database was planned considering performance and efficiency, and repeated storage of the same data was avoided. Data can be visualized and queried from the GH interface by using dedicated GH components, developed to accommodate required functionalities. An example application of the developed components was evaluated through the definition of GH component sequences. Each sequence is constructed to facilitate a specific functionality, and the intended functionality is tested by the designer. Furthermore, the designer can filter the data contained in the Voxel Model by executing a predefined SQL query and interactively evaluate the outcomes of the

query. Lastly, multi-scalar voxel model data can be transformed into a graph-representation that can be queried through the GraphDB endpoint. In result, simple SPARQL-based reasoning queries can be executed directly on the data contained in the ECOLOPES Voxel Model. Figure 5 shows the different levels in the ECOLOPES Voxel Model that facilitate interactions between different processes implemented within the GCD process.

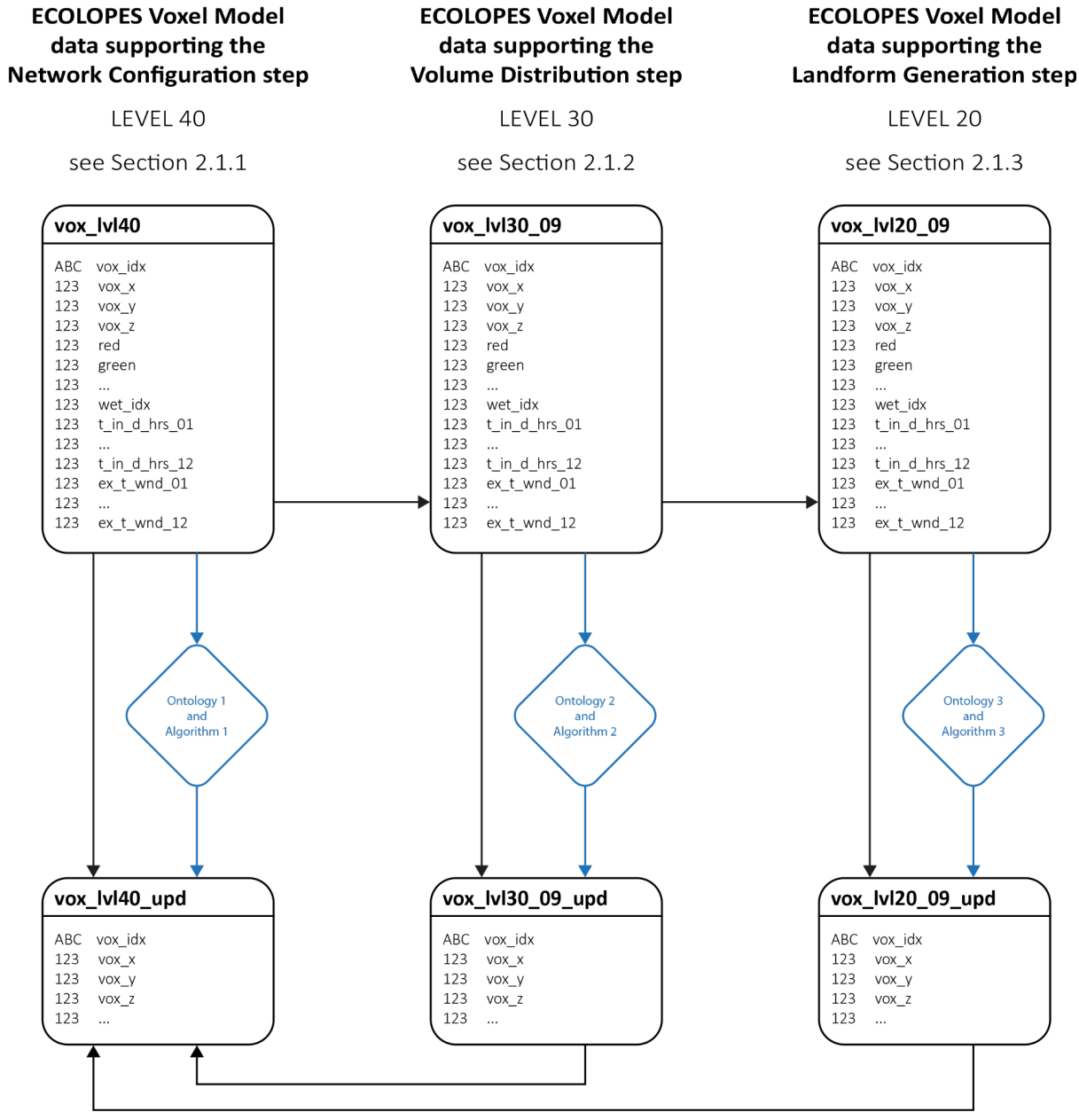

Fig. 5: Overview of the selected levels contained in the ECOLOPES Voxel Model and their relation to the computational procedures implemented in the three loops of the GCD process. Outcomes of each loop can be merged with the input data and written to a separate table (e.g., vox_lvl30_09_upd). This updated voxel-based representation can be merged with the large-scale data (vox_lvl40) and visualized in Rhinoceros.

To achieve the goal of multi-scalar data integration within the RDB environment, a site-aligned coordinate system was implemented. The role of the ontology-aided generative computational design process is to generate site-specific design proposals. The coordinate space of the data to be queried by EIM Ontology 2 and EIM Ontology 3 is aligned with the site boundary dimensions and the rotation of the site outline. These site-specific conditions can be described as a 2D rectangle with fixed dimensions and rotation. By extruding this 2D boundary by a fixed distance, a 3D bounding box that describes the site geometry is created. To support the operation of EIM Ontology 2 and EIM Ontology 3, an interactive reprojection of the RDB-based voxel model data from the original coordinate system to the site-aligned coordinate space was required. The implementation of geometric operations such as translation and scaling in the RDB environment is relatively straightforward. Inclusion of the rotation component was required since it would not be reasonable to limit the generative design process to operate exclusively on north-south oriented volumes for any given site. To include the rotation component in the internal transformation function executed in the RDB environment, custom SQL functions were developed. Those functions are used to map the coordinates between the large-scale voxel data and the site-scale coordinate system. Since the vertical location is not influenced by such a transformation, this problem is reduced to a composite transformation in 2D. In computer graphics, geometric transformations are conventionally utilizing homogeneous coordinates since all common transformation operations can be expressed as 3x3 matrices. In result, these operations can be combined by a simple vector multiplication operation. For this reason, the implemented transformation function required the voxel nodal point coordinates to be expressed as homogeneous coordinates. In homogeneous coordinate space a single point is expressed in a 3x1 matrix. Geometric operations of translation and rotation are represented as 3x3 matrices. The coordinate transformation needed to be implemented within the RDB environment. Since matrix operations are not implemented as a part of the SQL function syntax, coordinate transformation needs to be expressed as a system of linear equations.

For each required level a new SQL table view utilizing these functions was created. Finally, different site-aligned coordinate systems contain data in different resolutions. To facilitate this functionality, a simple aggregation function was used in the definition of the SQL table view. In result, datasets contained in the ECOLOPES Voxel Model can be queried in multiple resolutions and coordinate systems, both directly in the GH environment and indirectly through the EIM Ontologies implemented in the GraphDB environment.

The integration of the ECOLOPES Voxel Model and the EIM Ontology is facilitated through the Ontop Virtualization technology, which is part of the GraphDB software solution. The main task in such a virtualization-based workflow is related to the description of how the tables, columns, and primary keys in the RDB map onto graph structure in the GDB environment. On the technological level, such mapping is declared as a set of ODBA/R2RML mappings, written in a single file. This file is uploaded into the GraphDB instance when the RDB/GDB connection is created. This process has been tested to enable the integration between the RDB-based voxel model and EIM Ontology stored in the GraphDB environment. Initial tests showed that for performance reasons file-based SQL databases, such as SQLite, should be avoided. PostgreSQL and GraphDB server instances hosted on the same machine have shown sufficient performance to execute the operations required for the integration between the ECOLOPES Voxel Model and the EIM Ontology. Currently, each voxel model level contained in the RDB

can be interactively linked with the GraphDB instance, based on the provided OBDA/R2RML mappings. As a result, the data contained in the RDB-based voxel model can be transparently queried and reasoned using techniques implemented in the GDB environment.

The interactions between ECOLOPES Voxel Model, designer and the GCD components are divided into three loops.

Loop 1 facilitates the configuration of *networks* in a voxelized 3D space, materialized in the Rhinoceros CAD environment. It involves EIM Ontology 1, Rule-Based System 1 (associated algorithms i.e., ASP), and CAD Model 1 as its main components of interaction in the Translational Design Process of selection and distribution. Defining and spatially locating design objectives and implementing design instructions derived from EIM Ontology 1, based on feedback from Loop 2 and Loop 3, are some of the tasks that are addressed.

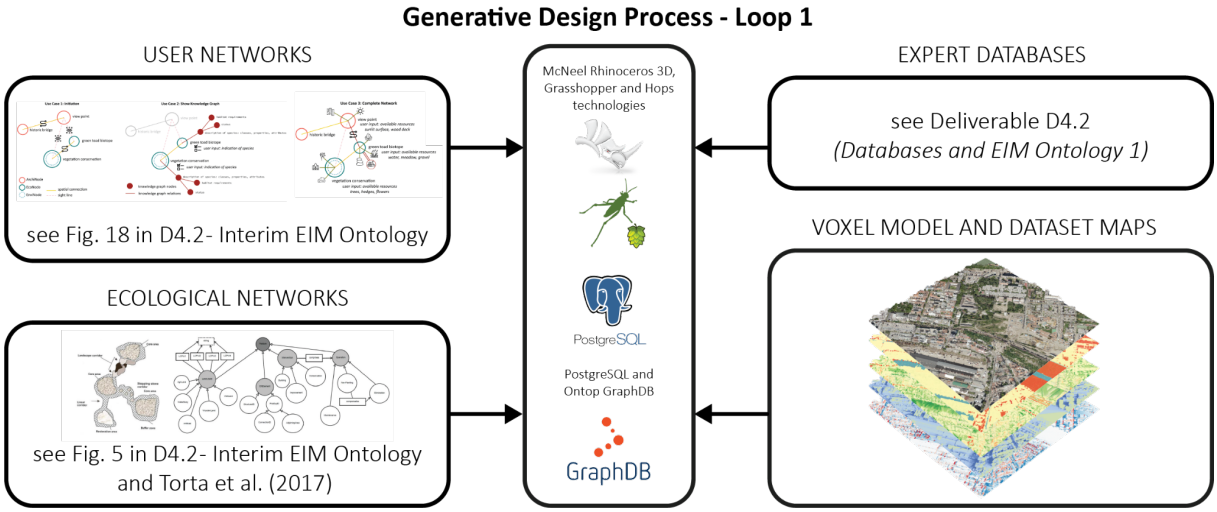

Fig. 6: Loop 1 of the generative design process.

Loop 2 facilitates the distribution of Volumes in a voxelized 3D space, materialized in the Rhinoceros CAD environment. It involves EIM Ontology 2, Rule-Based System 2 (associated algorithms i.e., ASP), and CAD Model 2 as its main components of interaction in the Generative Design Process_1 of volume specification. Guiding the changes in maps (volumetric voxel data) and 3D configuration of volumes, and implementing design instructions derived from EIM Ontology 2, based on feedback from Loop 1 and Loop 3, are some of the tasks that are addressed.

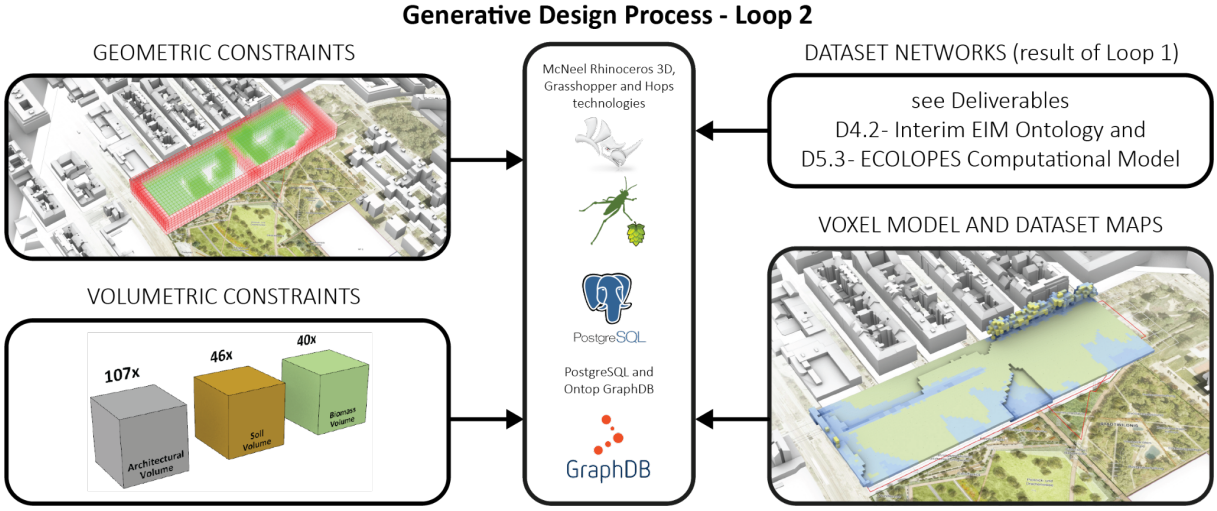

Fig. 7: Loop 2 of the generative design process.

Loop 3 facilitates the generation of *landform* geometry in a voxelized 3D space, materialized in the Rhinoceros CAD environment. It involves EIM Ontology 3, Rule-Based System 3 (associated algorithms i.e., ASP), and CAD Model 3 as its main components of interaction in the Generative Design Process_2 of landform generation. Guiding the geometric articulation of the volume object and implementing design instructions derived from EIM Ontology 3, based on feedback from Loop 1 and Loop2, are some of the tasks that are addressed. Detailed elaboration of the functionalities implemented within these three loops is presented in the report on *D4.2 Interim EIM Ontology*.

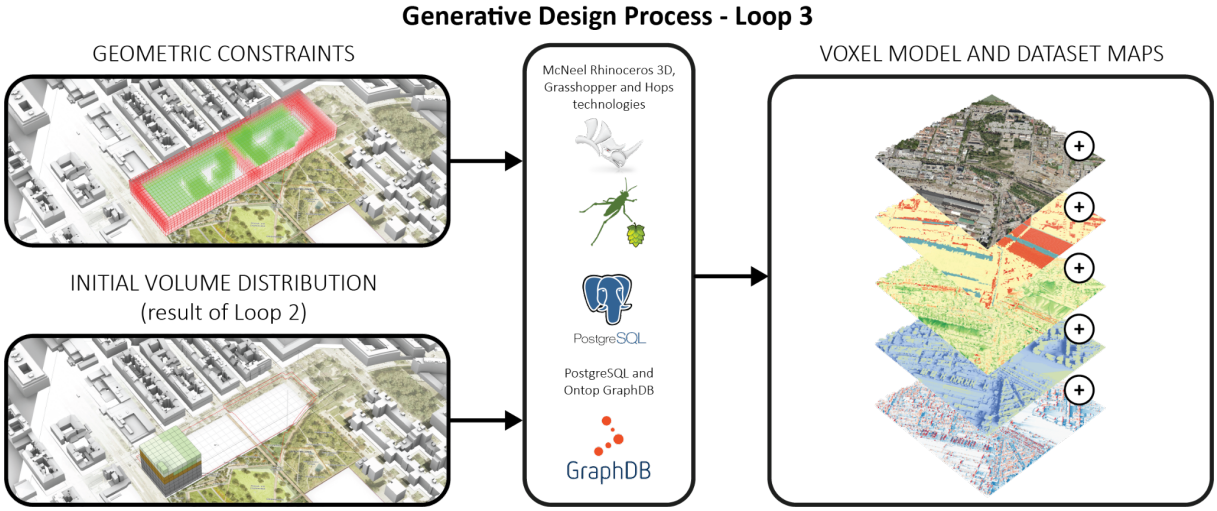

Fig. 8: Loop 3 of the generative design process.

Architectural designers can interact with the ECOLOPES Voxel Model both directly through the dedicated GH interface and indirectly, while working with other GCD components which are accessing ECOLOPES Voxel Model data in background. Different GCD components are implemented in the three loops, as described in the Deliverables *D4.2 Interim EIM Ontology* and *D5.4 ECOLOPES Computational Model*.

2 ECOLOPES Voxel Model Demo

The datasets contained in the voxel model can be utilized in the processes implemented within the optimization workflow (WP6). At the end of the process implemented in the WP4 / WP5 workflow, both CAD-based and voxel-based representation of the geometry will be saved and passed to the components positioned later in the ECOLOPES Computational Design Workflow (Fig. 1). The results of the GCD process will be exported as a 3D CAD model in the Rhinoceros format (.3dm). Detailed description of the outcomes generated in the GCD process is described in D5.3 ECOLOPES Computational Model. Corresponding voxel-based data will be exported as a single SQLite database file, containing domain-specific data related to the environmental and ecological properties of the design outcome at the end of the ontology-aided generative computational design process. This numerical information can be computed as fitness objectives for the multi-objective optimization. As described in the sections above, these fitness objectives are the KPIs that have been computed in the KGF (WP3). Additionally, the ontological correlations will aid in the definitions of KPI priorities and design strategies. The resulting optimized 3D design alternatives and metadata could also be appended into the database file to enable an iterative process between the generative and optimization design phases.

ECOLOPES Voxel Model combines a range of widely used technologies and diverse methods of combining individual components have been investigated throughout the project. The following description outlines an approach that foregrounds reproducibility by utilizing easily available software tools. Such workflow is not optimized for performance or automatization of repeatable tasks. This description is meant for demonstrative purposes and for this reason the amount of detailed descriptions of minor functionalities has been reduced.

2.1 Structure of the ECOLOPES Voxel Model

The structure of the ECOLOPES Voxel Model is open-ended and the choice of the datasets included is informed by the intended application of the data in the design task. Currently, an initial selection of a few datasets has been made to probe the affordances of our approach, as well as to guide the development of the ECOLOPES Voxel Model components.

The ECOLOPES Voxel Model has been developed to integrate data in multiple scales and to enable interoperability between the datasets. This functionality has been developed by introducing the following concepts into the ECOLOPES Voxel Model implementation: Design operations are carried out in a user-defined scale, for which a resolution and a spatial extent is chosen. For such fixed configuration, we use the term ‘voxel level’. Data contained in a single voxel level, contained in the ECOLOPES Voxel Model, is organized, based on a unique value, and referred to as the voxel (vox_idx). In technical terms, voxel index is at the same the primary key of the SQL table storing data contained in each voxel level. As the underlying data is prepared for a single voxel level, all properties are calculated for 3D, spatial coordinates that can be represented with integer numbers. It needs to be noted that the physical size of the voxel cells is calculated by multiplying those coordinates with the base resolution of the voxel level. This allows the voxel model to represent datasets in arbitrary resolutions. The values stored as voxel index are created by concatenating the three coordinates of the integer, spatial

coordinates. For example, a voxel cell located at vox_x = 1, vox_y =2 and voz_z = 3 would be indexed with a “1_2_3” voxel index value. Each voxel cell contains the voxel index property, the three voxel nodal point coordinates (vox_y, vox_y, vox_z) and an arbitrary number of additional properties. In a tabular view, those properties are represented as additional columns in the table. Each of those properties can be visualized as a false-color image in the 3D environment of the Rhinoceros 3D software. In this representation, visualization of the additional voxel model properties looks like multiple layers of a 3D map. For this reason, the name ‘data layers’ has been chosen to refer to the additional properties contained in the SQL tables representing individual levels within the ECOLOPES Voxel Model.

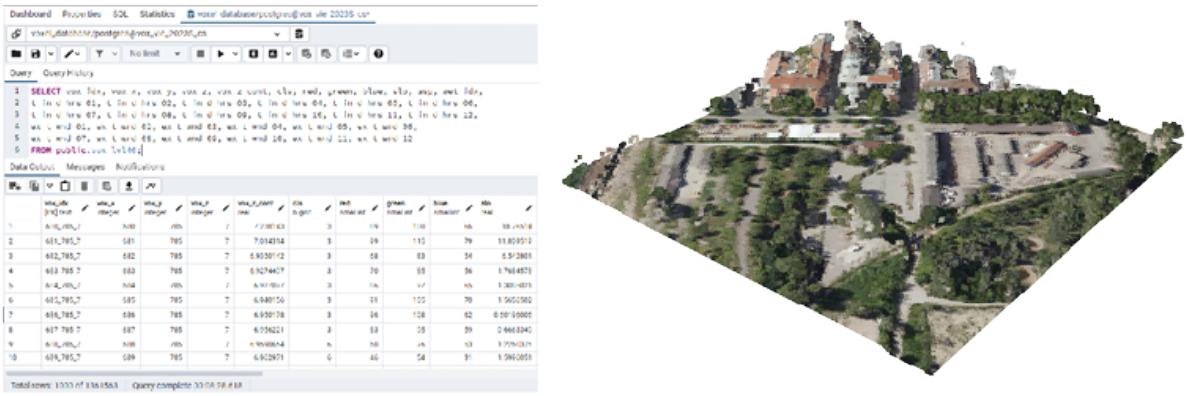

Fig. 9. Left: Tabular representation of the ECOLOPES Voxel Model data that shows the different columns that represent individual datasets encoded within the voxel model. Right: 3D representation of the exemplary data layer generated with the implemented Rhinoceros 3D / Grasshopper components.

Datasets stored in the individual voxel levels are representing different fragments of the real-world location in which the design task will be executed. As described in the previous paragraph, those fragments may vary in spatial resolution and the extent which they cover. To ensure the correct location of those fragments, their relative location must be recorded. For this reason, the vox_meta SQL table has been introduced. This table stores information about the geometric location of the individual fragments, as well as their absolute location in a standardized geographic coordinate reference system. Additionally, information describing the resolution and rotation of individual voxel levels is recorded in this table.

2.2 Structuring Spatial Data for Inclusion in the ECOLOPES Voxel Model

For demonstration purposes an exemplary raster data layer was selected and loaded into QGIS. In this example we use Digital Surface Model (DSM) data which stores height information. We generate the voxel nodal point coordinates (vox_x, vox_y, vox_z), as well as an absolute height value (vox_z_cont) for each cell in this reduced 2.5D version of our temporary voxel data. The easiest way to export raster data from QGIS into a .csv file is the standard gdal2xyz tool. The generated file will contain three columns, the first two containing spatial coordinates of all points in the raster and the third containing the values contained in the raster data. Spatial coordinates are represented in the geographic coordinate reference

system, which needs to be set up in QGIS. In this simple example we are loading the .csv data into Excel and referencing the data on a separate sheet for the export to the ECOLOPES Voxel Model. In the second Excel sheet, basic data formatting tasks will be implemented. For example, the vox_idx column is generated with a default Excel function as a concatenation of the vox_x, vox_y and vox_z fields. The exported .csv file needs to be sent to the PostgreSQL server and loaded into the required table (vox_lvl00 in this example). Figure 10 presents the steps described above.

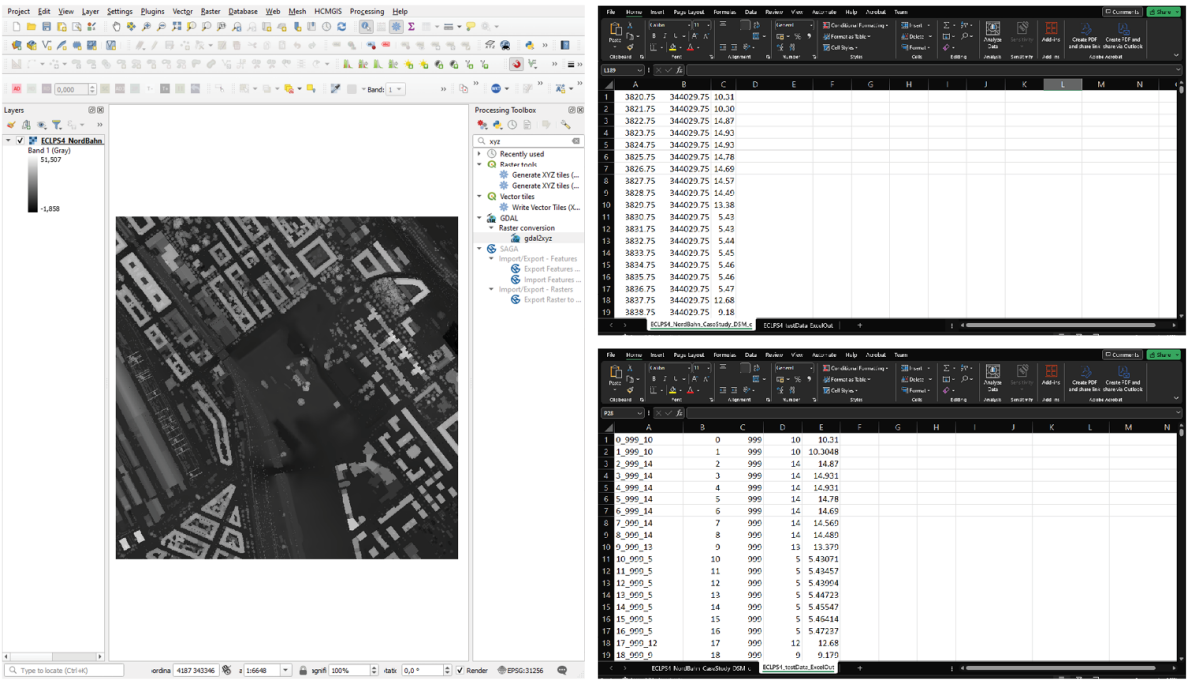

Fig. 10: Left: Exemplary DSM data loaded into QGIS interface. Right: Previews of the two Excel sheets in which the data conversion is performed.

In this example, the raster data was loaded from a widely used raster data format (GeoTIFF) using basic functionalities provided by QGIS and Excel software. Theoretically, multiple raster files could be exported as layers and merged into a single .csv file and exported to the ECOLOPES Voxel Model. Such a process would require user supervision, since the data conversion process requires at least sanity checking at each step. While preparing input raster data for the ECOLOPES Voxel Model, basic understanding of standard coordinate reference systems used in the GIS systems is also required. Lastly, the initial .csv file generated with QGIS is 85% larger than the raster-based equivalent. For this reason, such simplified workflow is not efficient for large datasets. The final step in this part of the demonstration was the visual checking of the loaded data in the Rhinoceros interface. Figure 11 shows the data loaded from QGIS in this example, rendered as a 3D representation of the data contained in the ECOLOPES Voxel Model.

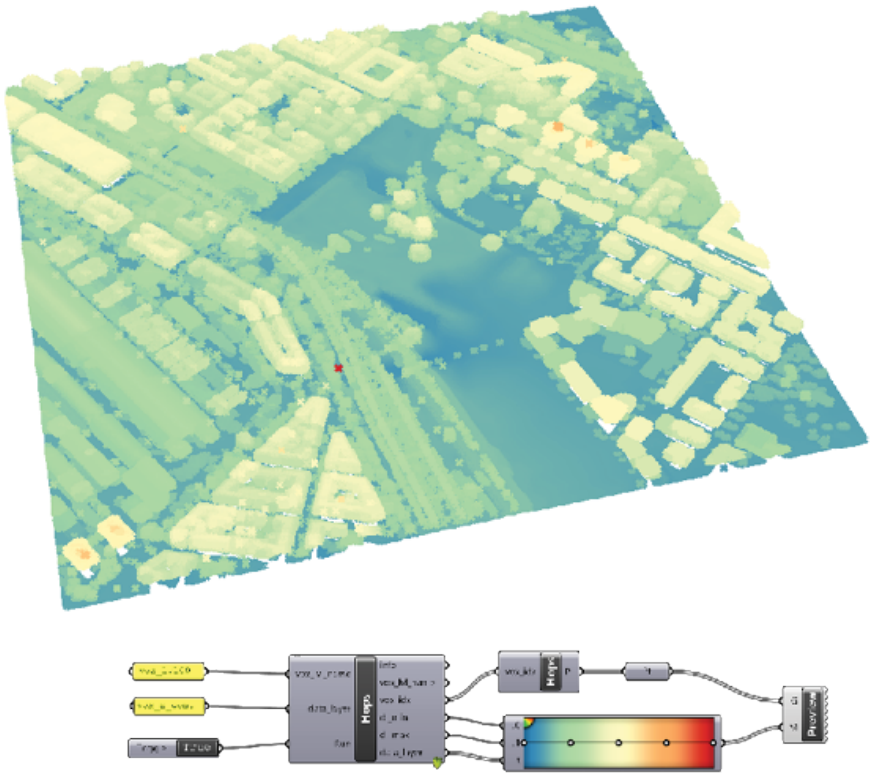

Fig. 11: Voxel data generated in this example loaded into Rhinoceros 3D interface by utilizing the developed ECOLOPES Voxel Model components. Colors represent the absolute height of the points (vox_z_cont layer loaded from the DSM).

2.3 **Loading data into the ECOLPES Voxel Model**

The most straightforward method to load a medium-sized dataset into an SQL database involves defining individual tables and filling it with data from the provided .csv file. SQL tables can be created with the CREATE TABLE statement followed by the definition of required columns and data types. Contents of a .csv file can be copied into the previously created table in the PostgreSQL environment using the COPY statement.

SQL servers that are utilizing the client-server model are often installed on a separate server with limited physical access. By default, PostgreSQL contains the psql command-line utility to allow administrative operations on the server instance. SQL statements, such as the CREATE TABLE and COPY statements can be executed with the use of psql utility. Although this method might be considered very rudimentary, it displayed the highest performance in copying medium-sized large datasets into the PostgreSQL instance. Properly formatted CSV data can be written from Excel or using Python pandas library and uploaded in parallel chunks onto the server storage. Then the psql utility can process the local file avoiding network latency constraints.

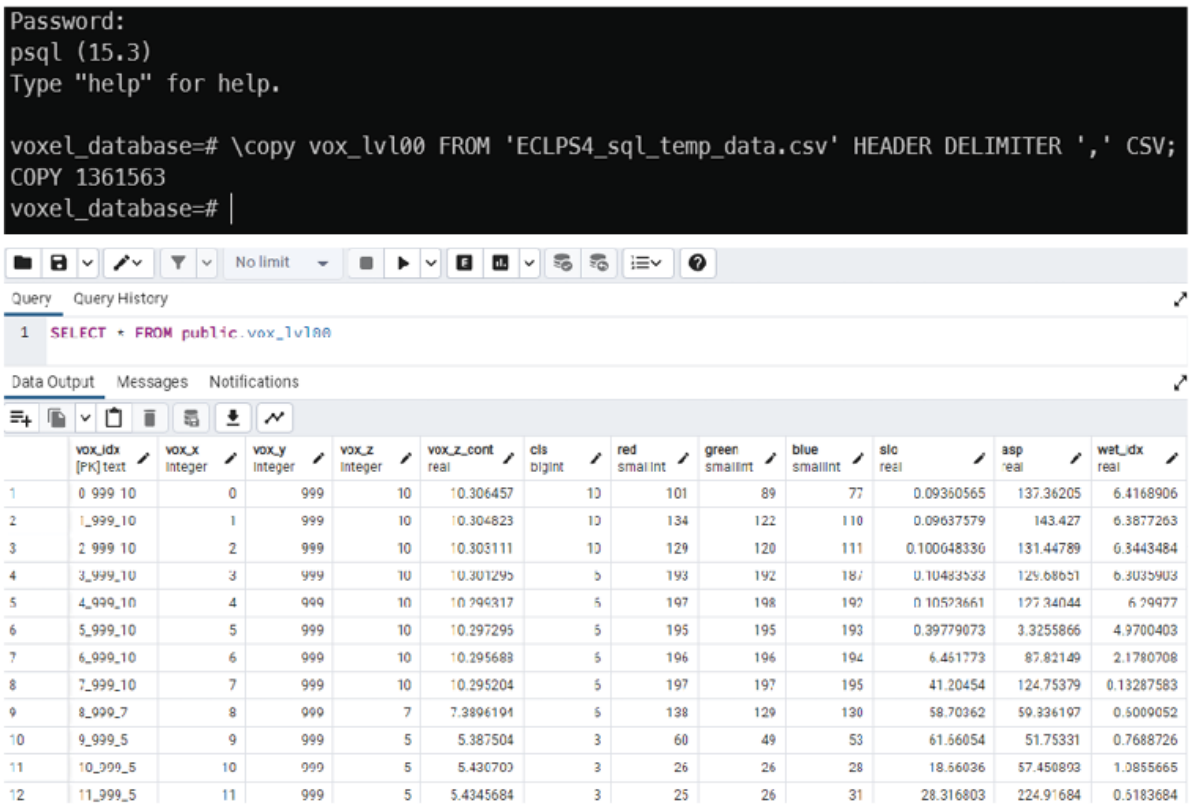

Fig. 12: Top: Required psql query which allows very fast importing of csv data into the remote PostgreSQL instance. Bottom: Data loaded with the psql tool can be visually inspected in the pgAdmin interface.

Required conversions and the final formatting of the .csv file can be done using any software tool which is familiar for the end user. The use of such widely adopted data formats would allow the integration of datasets created with external tools, such as the ECOLOPES Ecological Model. Initial tests on very small datasets can be initiated by sending the prepared .csv file into the online PostgreSQL instance with a web-based administration interface, such as pgAdmin. The following section shows how the data contained in the ECOLOPES Voxel Model can be browsed in the pgAdmin interface.

2.4 Default Web-based Interface for ECOLOPES Voxel Model Data (pgAdmin)

A widely used web interface for PostgreSQL databases is the open-source pgAdmin software. This software allows for interactive and user-friendly administration of a PostgreSQL server running externally (e.g., in the cloud). It exposes most of the functionalities of a PostgreSQL server in a graphical interface. An overview of the functionalities most relevant for the operation of ECOLOPES Voxel Model is presented in Figure 13.

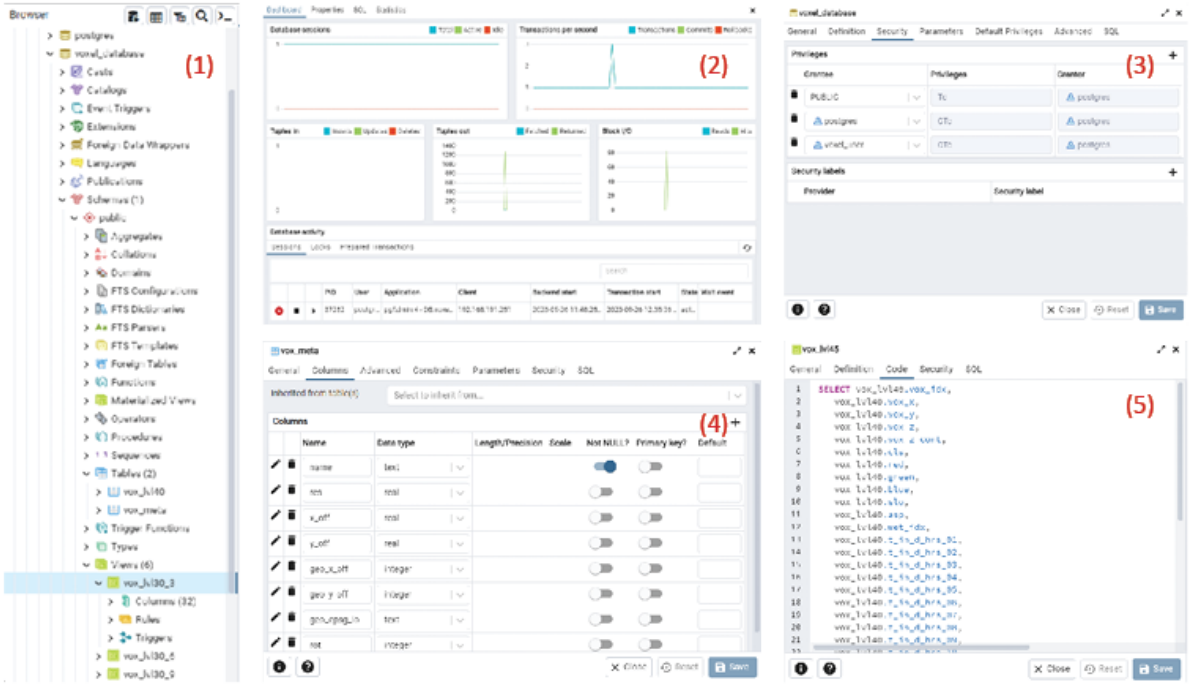

Fig. 13: Overview of the functionalities available in the pgAdmin interface. Browser window (1) showing a hierarchical overview of contents of the SQL database. Dashboard (2) provides basic metrics related to the performance of the running queries and opened sessions. In the security section (3) fine grained user permissions can be set. Contextual preferences menu allows to modify defined table columns (4) or update the SQL code that defines a named view (5).

2.5 Rhinoceros / Grasshopper Interface for the ECOLOPES Voxel Model

Data representation offered by the pgAdmin application is not well suited for any design activity, because it is difficult to identify spatial patterns in data represented in a table. For this reason, an interface between the ECOLOPES Voxel Model based on the PostgreSQL technology and the Rhinoceros / Grasshopper software had to be created. In the following paragraphs individual functionalities of the implemented components are described.

2.5.1 Listing available voxel levels in the ECOLOPES Voxel Model

The first required component was implemented to list the available voxel levels from the ECOLOPES Voxel Model. This very simple functionality works as an entry point for all scripts that are utilizing the ECOLOPES Voxel Model GH interfaces. As the development of the components progressed, a need to filter out a few SQL tables appeared and this component was used to facilitate this functionality. Figure 12 shows the use of this component. Simple boolean toggle (switch button) need to be connected to this component to initiate the computation. All implemented GH components are providing basic debug information in the “info” outputs. This component outputs the prefiltered list of available voxel levels. The remaining components shown in Figure 14 are standard GH components to select a single item and to preview the contents of a list.

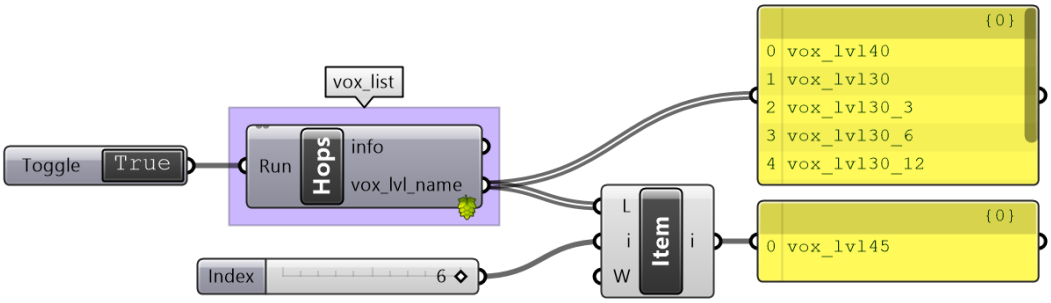

Fig. 14: Voxel levels available in the ECOLOPES Voxel Model can be retrieved using a dedicated Grasshopper component.

2.5.2 Retrieving metadata from the ECOLOPES Voxel Model

A dedicated component for retrieving metadata describing the key parameters of the individual voxel levels has been implemented. This component requires the name of a voxel level as an input. The vox_lvl_name property is passed through the component to simplify the layout of the multiple components and informs the data flow inside the algorithms represented by the GH definition. Resolution, rotation, relative and absolute offsets are returned by this component (see Section 2.1). These outputs are used in multiple components positioned later in the exemplary Grasshopper scripts.

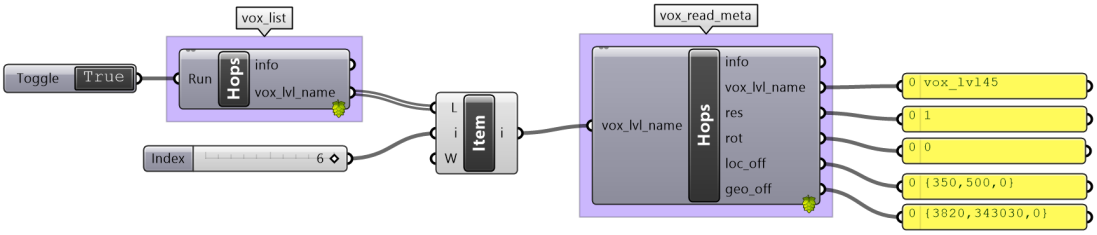

Fig. 15: Metadata describing individual levels contained within the ECOLOPES Voxel Model can be retrieved inside the Grasshopper environment.

2.5.3 Retrieving the voxel index data (vox_idx) and geometric data from the ECOLOPES Voxel Model

We implemented a GH component that enables the designer to retrieve the voxel index from a single voxel level. This functionality allows the designers to use conventional GH techniques for data checking, sorting and filtering with the data retrieved from the ECOLOPES Voxel Model. Currently the geometric data is retrieved with the same component. Geometry returned by this component is formatted as native Rhinoceros 3D point objects and can be directly visualized using readily available tools. This component requires the voxel level name as an input. Since a large query can take relatively long time to execute, a boolean toggle

(switch button) input has been added. This requires users to make a conscious decision to run the query. This component shown in Figure 16 returns the voxel index values, voxel nodal coordinates as Rhinoceros points and a collection of 3D vectors that represent the real-world colors of the geometry. This functionality related to full-color visualization is described in the following paragraph. Point geometry returned by this component can also be moved to the local coordinate system to match the location of other levels contained in the ECOLOPES Voxel Model. Alternatively, the geometry can be moved to the absolute location in the selected, standardized geographic coordinate system. Standard Grasshopper components for the addition of vectors and translation of geometry are used in this example.

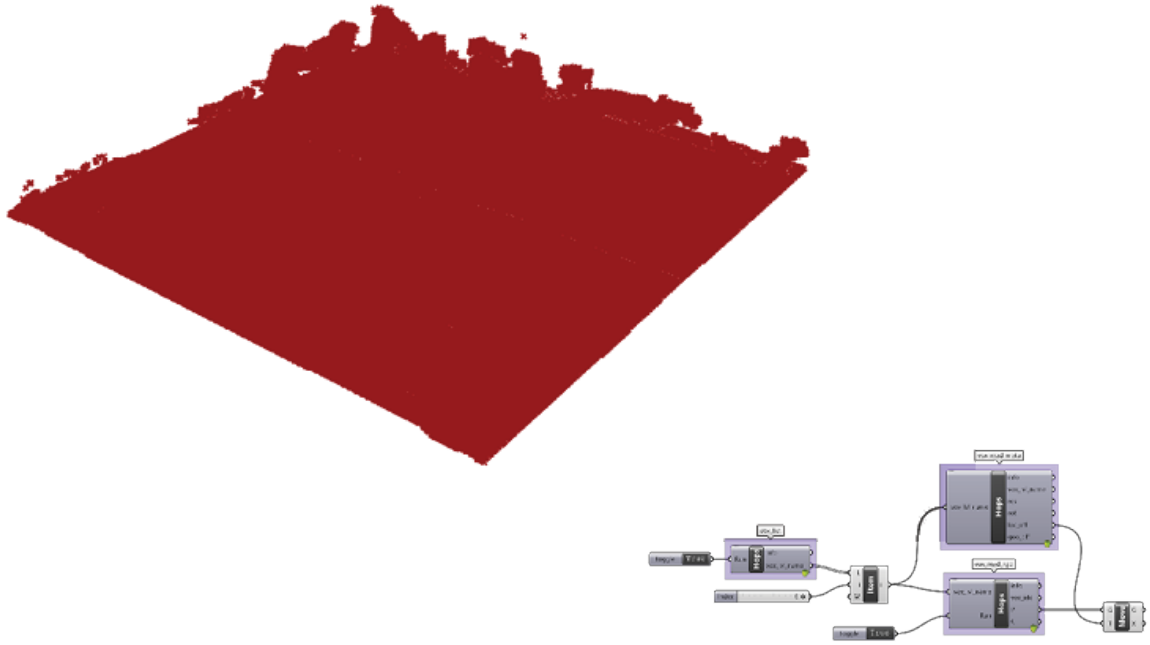

Fig. 16: Visual representation of the geometric data retrieved from the ECOLOPES Voxel Model with the implemented Grasshopper component.

2.5.4 Retrieving color data from the ECOLOPES Voxel Model

To improve the perception of the geometric data contained in the ECOLOPES Voxel Model, the real-world colors of the objects contained in the ECOLOPES Voxel Model can be retrieved. Currently the same component is used for retrieving geometry and real-world colors. The colors are retrieved as a list of 3D vectors and standard GH components are used to construct the instances of GH colors (“Deconstruct Vector” and “Colour RGB” components). This native representation of Grasshopper colors can be used with the Grasshopper Preview component to create the illustration presented in Figure 17.

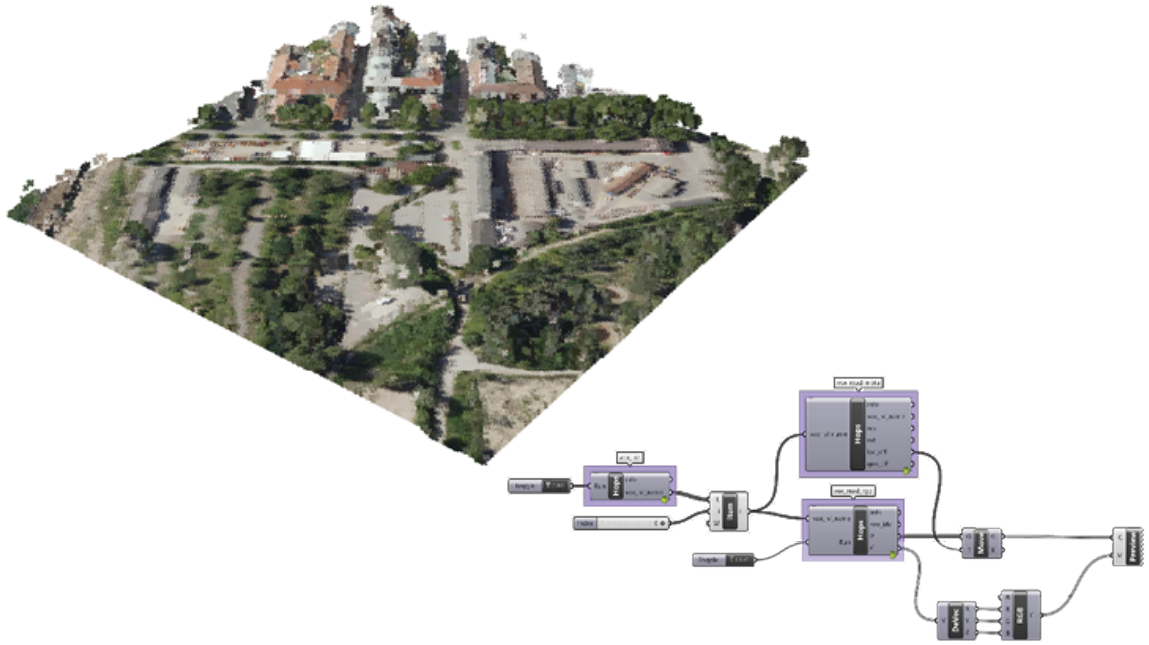

Fig. 17: Graphic representation of the geometric data supplemented with real-world colors retrieved from the ECOLOPES Voxel Model with the implemented Grasshopper component.

2.5.5 Retrieving the list of available data layers from the ECOLOPES Voxel Model

Additional data layers stored in each of the voxel levels need to be retrieved within the Grasshopper environment. To make this possible, firstly the available data layers need to be listed. Dedicated components have been required for this functionality, since available data layers differ between the voxel levels. Moreover, standard properties, such as the geometric coordinates and the voxel index need to be filtered out before the listing of available data layers. The component implemented for the retrieval of available data layers requires only the name of the voxel level as an input. It returns the pre-filtered list of the data layers available in the ECOLOPES Voxel Model.

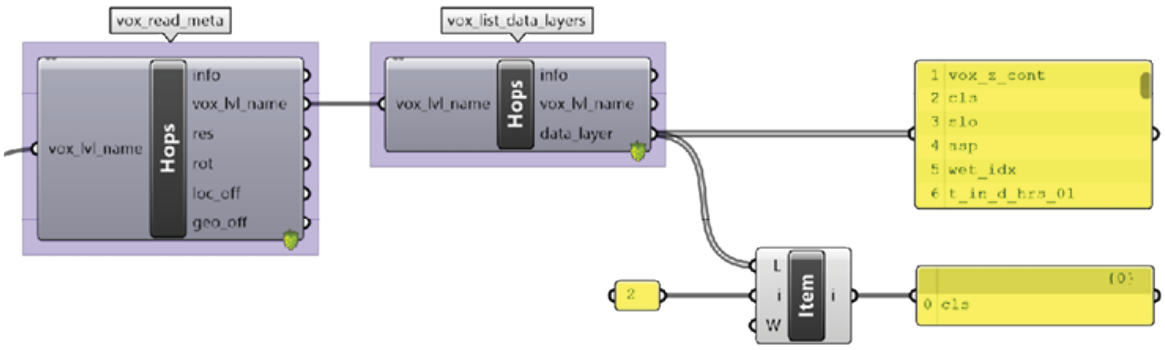

Fig. 18: List of available data layers retrieved from the ECOLOPES Voxel Model, presented as a list in the Grasshopper interface.

2.5.6 Retrieving a single data layer from the ECOLOPES Voxel Model

Besides retrieving the geometry and color data, each data layer can be retrieved from the ECOLOPES Voxel Model and manipulated in the GH environment. Standard GH components can be used to generate false color visualizations which can be interactively viewed in the Rhinoceros viewport. Components implemented for the retrieval of the data contained in a single data layer require the name of selected voxel level and data layer, which are returned by the components described in the previous paragraphs. Voxel index data and the numeric values of the chosen data layer for each voxel nodal point in the ECOLOPES Voxel Model are returned by this component. Additionally, the maximum and minimum value of the selected voxel data layer is provided. Those outputs are prepared to match standard GH components which allow for false color visualization. GH Gradient component and GH Preview component were used to create the illustration presented in Figure 19.

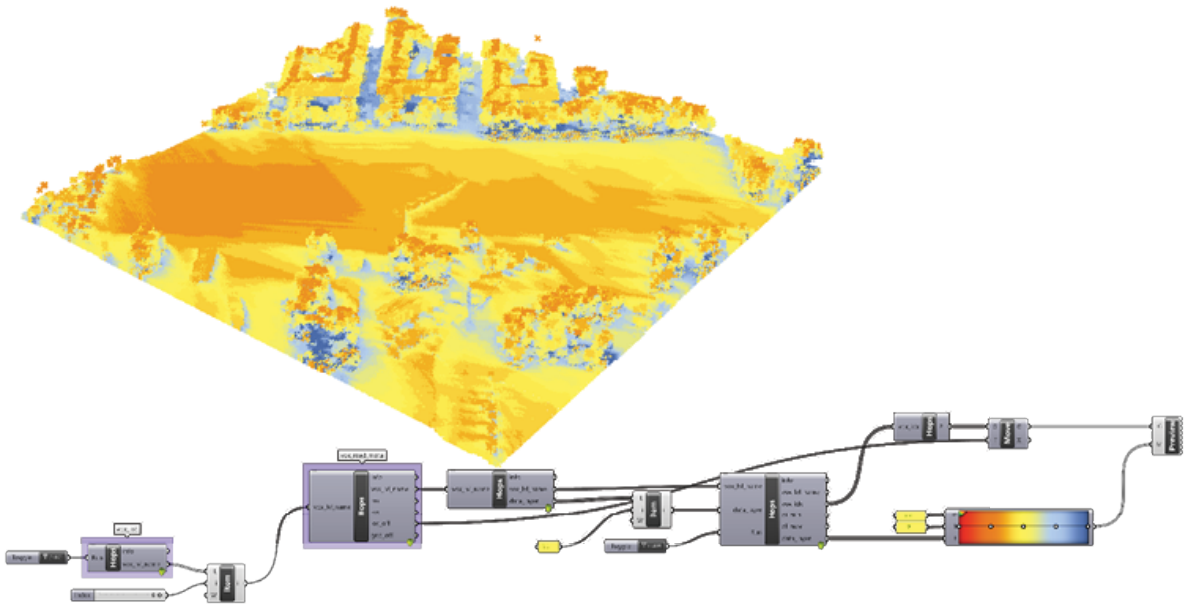

Fig. 19: Graphic representation of the selected data layer, presented with false colors in the Rhinoceros 3D interface. This representation is created using the ECOLOPES Voxel Model Grasshopper component presented on the right side of this figure.

2.6 Inserting new Geometry into the ECOLOPES Voxel Model

Different approaches for converting Rhinoceros geometry into a grid-based representation can be utilized to prepare the Rhinoceros geometry to be exported to the ECOLOPES Voxel Model. Diverse experiments have been executed within the ECOLOPES consortium and the simplest method is presented in this deliverable, with the aim to showcase the most reproducible approach.

2.6.1 Simplified method for exporting Rhinoceros 3D geometry to the ECOLOPES Voxel Model

For demonstration purposes an exemplary Grasshopper script to export closed solid geometry from GH to the ECOLOPES Voxel Model was developed. The script presented below does not require any external Grasshopper plugins and is expected to serve as a starting point for the further development of new components that will be utilizing the interface with the ECOLOPES Voxel Model and the EIM Ontology.

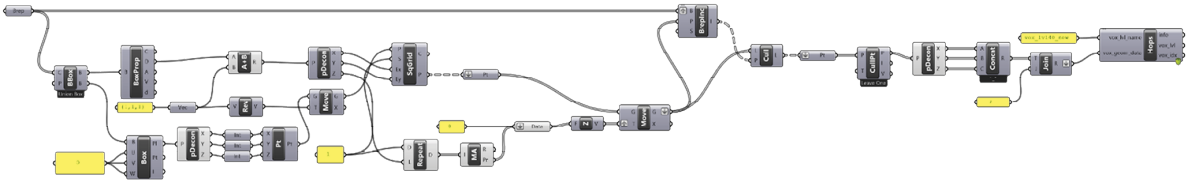

Fig. 20. This simple Grasshopper script allows for converting closed Rhinoceros geometries into a 3D grid-based representation for integration with the ECOLOPES Voxel Model.

The component of the GH script presented in Figure 20 exports the converted GH geometry to the ECOLOPES Voxel Model. This component requires properly formatted geometry and the name of the voxel level as inputs. Voxel index values of inserted points and the used voxel level are returned as outputs.

2.7 Exporting Data from the ECOLOPES Voxel Model

As in the case of loading data into the ECOLOPES Voxel Model, this demonstration describes the most straightforward method to export a medium-sized dataset from the ECOLOPES Voxel Model stored in the PostgreSQL database. In this example, the pgAdmin utility described previously in this deliverable will be used to retrieve the table-based representation of the data contained in the ECOLOPES Voxel Model. Exported data will be loaded back in QGIS software to showcase the interoperability of the implemented approach.

2.7.1 Exporting simplified 2.5D geometry from the ECOLOPES Voxel Model

In this example, an exemplary collection of closed geometries modeled in Rhinoceros are exported into the ECOLOPES Voxel Model using the script presented in the previous section. The chosen geometry shows that this method can process multiple objects simultaneously considering 3D features of the geometry, such as undercut surfaces or vertical voids between individual objects. A preview of this geometry is shown on the left side of Figure 21. As an outcome, this geometry will be exported to the ECOLOPES Voxel Model and the merged representation will be exported to a widely used raster format compatible with QGIS. The exported geometry is marked with a red circle on the right side of Figure 21.

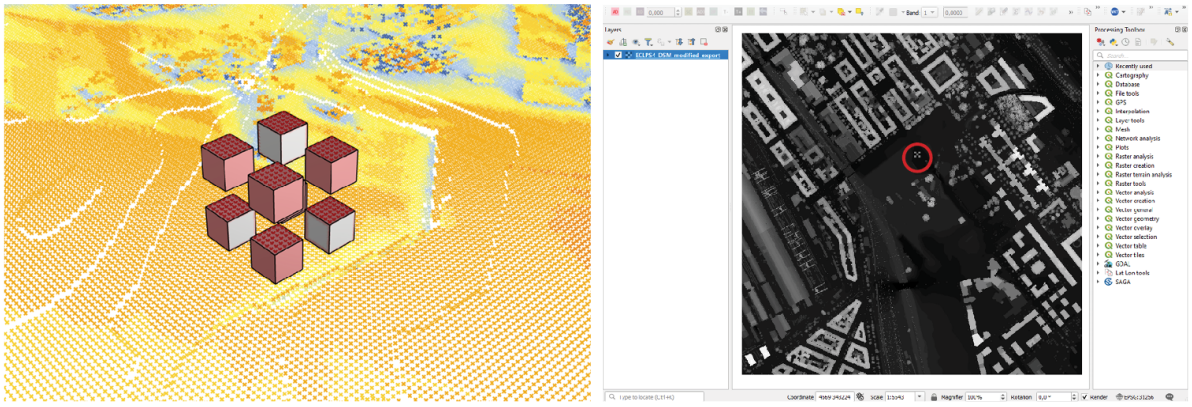

Fig. 21: Left: Exemplary collection of closed geometries modeled in Rhinoceros 3D. Right: The same geometry exported in raster format into QGIS.

The query used to retrieve the ECOLOPES Voxel Model data is shown in Figure 22 below. The presented query returns the data structured according to the standards required by common GIS software. It allows the user to add the offset between the local coordinate space of the ECOLOPES Voxel Model and the global coordinate reference system required by the GIS software. Columns exported from the SQL database are set to "x", "y" and "z", because the QGIS import function expects the names to match these values. This query utilizes a GROUP BY functionality preceded by the MAX function. This generates a 2.5D representation of the 3D voxel geometry in which only the topmost point at each pair of the 2D coordinates is returned. Lastly, to assure that the exported file works, the points need to be sorted first according to the "x" coordinate in the ascending order and secondly according to the "y'' coordinate in the descending order. This is facilitated by the two ORDER BY clauses and it results in a row-based data structure with the first point in the raster located in the upper left corner.

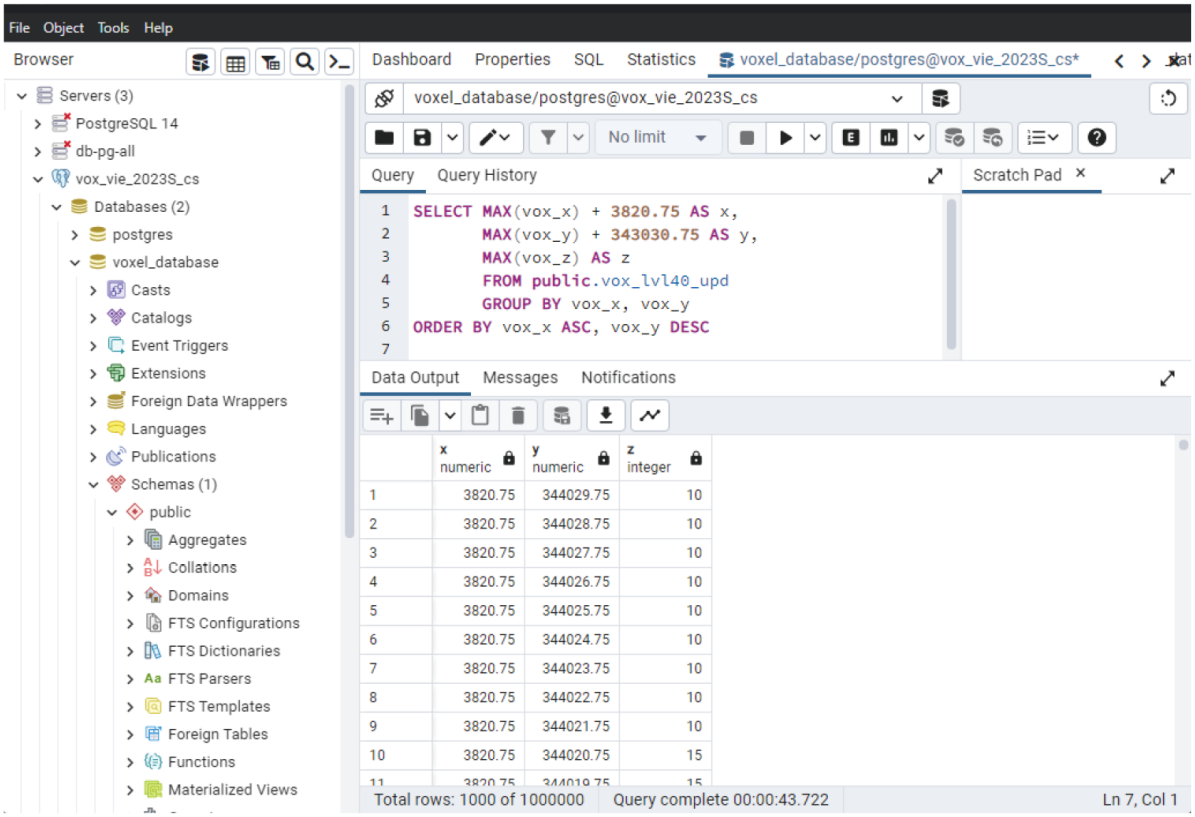

Fig. 22: Query used to export the ECOLOPES Voxel Model data into the .xyz format supported by QGIS. Generated file can be easily downloaded from the pgAdmin interface.

2.8 Integrating the ECOLOPES Voxel Model with the EIM Ontology

ECOLOPES Voxel Model and the EIM Ontology are implemented by utilizing different technologies, because of the different character of the ontological RDF data and the voxel data represented as SQL tables. Integration between the ECOLOPES Voxel Model and the EIM Ontology is facilitated by the Ontop Virtualization technology implemented in the GraphDB environment. This technology allows linking the data contained in the SQL database with the GraphDG environment. To establish this link, a short .obda file describing the mapping between the structure and datatypes of the SQL database and the structure and datatypes of the GraphDB data needs to be prepared. Example of such a file prepared to map a single voxel level is presented in Figure 23.

```
[PrefixDeclaration]
:           http://resource.dap.tuwien.ac.at
schema:     http://schema.dap.tuwien.ac.at#
ex:         http://example.org/
owl:        http://www.w3.org/2002/07/owl#
rdf:        http://www.w3.org/1999/02/22-rdf-syntax-ns#
xml:        http://www.w3.org/XML/1998/namespace
xsd:        http://www.w3.org/2001/XMLSchema#
foaf:       http://xmlns.com/foaf/0.1/
obda:       https://w3id.org/obda/vocabulary#
rdfs:       http://www.w3.org/2000/01/rdf-schema#

[MappingDeclaration] @collection [[

mappingId   lvl30
target      :/vox/{vox_idx} a schema:Voxel ; schema:x {vox_x}^^xsd:integer  ; schema:y {vox_y}^^xsd:integer  ; schema:z {vox_z}^^xsd:integer  ;
schema:zCont {vox_z_cont}^^xsd:decimal  ; schema:isNew {is_new}^^xsd:integer  ; schema:cls {cls}^^xsd:integer  ;
schema:red {red}^^xsd:integer  ; schema:green {green}^^xsd:integer; schema:blue {blue}^^xsd:integer  ;
schema:slope {slo}^^xsd:integer  ; schema:aspect {vox_z}^^xsd:integer  .
source      SELECT * FROM vox_lvl30;

mappingId   lvl40
target      :/vox/{vox_idx} a schema:Voxel ; schema:x {vox_x}^^xsd:integer  ; schema:y {vox_y}^^xsd:integer  ; schema:z {vox_z}^^xsd:integer  ;
schema:zCont {vox_z_cont}^^xsd:decimal  ; schema:isNew {is_new}^^xsd:integer  ; schema:cls {cls}^^xsd:integer  ;
schema:red {red}^^xsd:integer  ; schema:green {green}^^xsd:integer; schema:blue {blue}^^xsd:integer  ;
schema:slope {slo}^^xsd:integer  ; schema:aspect {vox_z}^^xsd:integer  .
source      SELECT * FROM vox_lvl40;


]]
```

Fig. 23: Exemplary OBDA file that defines the mapping between ECOLOPES Voxel Model and the EIM Ontology. This file is loaded in the GraphDB environment to enable integration between the different representations of data utilized by the ECOLOPES Voxel Model and the EIM Ontology.

GraphDB platform stores data as RDF triples, which is the most common information representation in ontologies. EIM Ontology is implemented in the GraphDB environment and the ECOLOPES Voxel Model data is automatically linked with its RDF-based representation in GraphDB. This allows the execution of typical ontological operations to be executed on the data linked from the ECOLOPES Voxel Model. Figure 24 shows the basic SPARQL query being executed in the GraphDB environment which returns ECOLOPES Voxel Model data represented as triples.

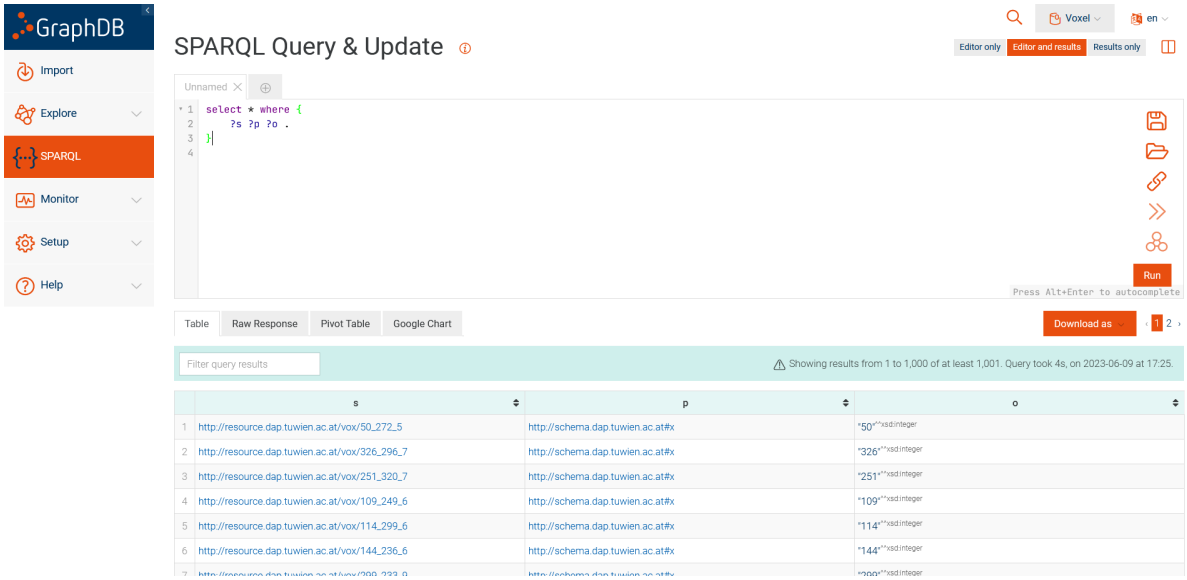

Fig. 24: Example of the SPARQL query which utilizes the OBDA Virtualization technique to return the triple-based representation of the ECOLOPES Voxel Model data.

2.8.1 Querying the ontological representation of the ECOLOPES Voxel Model data in Grasshopper

GraphDB software exposes a SPARQL endpoint allowing execution of standardized queries on the data stored in the database. In the case of ECOLOPES, this functionality is utilized to query the EIM Ontology, based both on the RDF data stored within the GraphDB and the virtualized ECOLOPES Voxel Model data. This functionality has been utilized to implement a direct interface between the ECOLOPES Voxel Model, EIM Ontology and the Rhinoceros / Grasshopper environment. An initial prototype of a GH component that allows querying the GraphDB data has been developed to showcase the possibility of querying the ontological representation of the ECOLOPES Voxel Model data. This functionality is showcased based on a simple example. In this simple example, suitable locations for 3 exemplary plant species are inferred based on the SPARQL reasoning technology and the virtualized ECOLOPES Voxel Model data. Results of the inference are fed back to the Rhinoceros environment and visualized using the methods implemented as a part of the ECOLOPES Voxel Model integration. Figure 25 illustrates this process in which the suitable locations for aHalfShadyPlant, aShadyPlant and aSunnyPlant are retrieved. This data can now inform the generative computational design process and be included in a data-package for the subsequent optimization process (WP6).

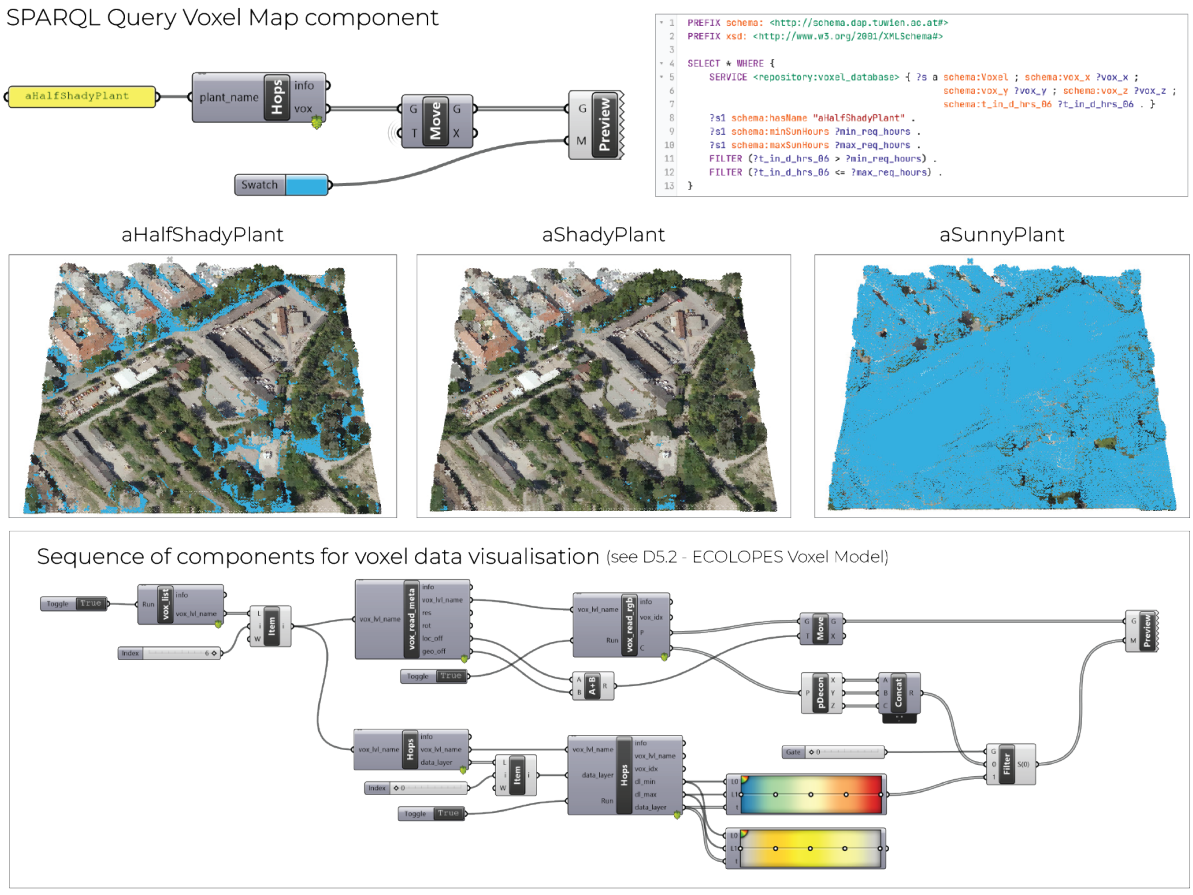

Fig. 25: Initial implementation of the Grasshopper component sequence allowing to execute SPARQL queries on the ECOLOPES Voxel Model data linked with the GraphDB environment.

3 PUBLICATION PLAN

We recently submitted a scientific article for peer-review to Frontiers of Architectural Research journal that focuses on the conceptual framework for an ontology-aided generative computational design process for ecological building envelopes. In the article we describe the conceptual approach and the development of the related components of the ontology-aided generative computational design process (EIM Ontologies, ECOLOPES Voxel Model, ECOLOPES Computational Model). Secondly, we are in the process of preparing a scientific article on the specific development and utilization of the "ECOLOPES Voxel Model as a *spatial-knowledge representation schemata* in the context of an ontology-aided generative computational design process for ecological building envelopes". We aim to submit this article to *Architectural Science Review* (Taylor & Francis) or to *Technology | Architecture + Design* (Taylor & Francis). Finally, we will prepare a scientific article on "Validation of the ECOLOPES ontology-aided generative computational design process for ecological building envelopes", which will report the results of the Vienna Case Study, which will commence in October 2023 and be finalized in January 2024.

REFERENCES

Conrad, O., B. Bechtel, M. Bock, H. Dietrich, E. Fischer, L. Gerlitz, J. Wehberg, V. Wichmann, & Böhner, J. (2015). System for Automated Geoscientific Analyses (SAGA) v. 2.1.4. *Geoscientific Model Development*, *8*(7), 1991–2007. https://doi.org/10.5194/gmd-8-1991-2015

Open Source Geospatial Foundation Project. "QGIS Geographic Information System," 2020. http://qgis.org.

Weisser, W., Hensel, M., Barath, S., Grobman, Y.J., Hauck, T.E., Joschinski, J., Ludwig, F., Mimet, A., Perini, K., Roccotiello, E., Schloter, M., Shwartz, A., Sunguroğlu Hensel, D., Vogler, V. (2022) Creating ecologically sound buildings by integrating ecology, architecture, and computational design. *People & Nature*, 5(1), 4-20. https://doi.org/10.1002/pan3.10411

Wilson, J. P., & Gallant, J. C. (Eds.) (2000). Terrain analysis: Principles and applications. Wiley.

SOURCE CODE AND SOFTWARE CITATION

Dragonfly-grasshopper. (2022). [Python]. Ladybug Tools. https://github.com/ladybug-tools/dragonfly-grasshopper

IronPython. (2023). [Python]. IronLanguages. https://github.com/IronLanguages/ironpython2 (Original work published 2017)

Robert McNeel & Associates. (2021). Ghhops_server: Grasshopper Hops Server (version 1.4.1). https://github.com/mcneel/compute.rhino3d/tree/master/src/ghhops-server-py

Robert McNeel & Associates. (2023). Rhino3dm [C#, Python]. Robert McNeel & Associates. https://github.com/mcneel/rhino3dm (Original work published 2018).

Rutten, D. (2023). Grasshopper (1.0.007) [C#; Windows]. https://www.grasshopper3d.com/

SQLAlchemy. (2023). [Python]. SQLAlchemy. https://github.com/sqlalchemy/sqlalchemy