ECOLOPES

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

H2020-FET-OPEN-2021-2025

Action number 964414

D3.1: Prototype technical requirements report

Dissemination level:	Public
Contractual date of delivery:	Month 12, 31 March 2022
Actual date of delivery:	Month 12, 29 March 2022
Work package:	WP3
Task:	T3.1, T3.2, T3.3 and T3.4
Type:	Report
Approval Status:	Final version for submission
Version:	v1.0
Number of pages:	52
Filename:	D3.1_PrototypeTechnicalRequirementsReport.docx

Abstract: The objective of D3.1 is to report on the current state of the work carried out in work package 3 (WP3) after the first year. Thus, besides a description of the technical requirements for the ECOLOPES platform, it presents the elaborated frameworks such as the computational framework (Section 3), the knowledge generation framework (Section 4), and the cloud-based Rhino.Compute framework (Section 5) to build the first prototype of the computational platform – the ECOLOPES sandbox (Section 6). Additionally, to contextualise the ECOLOPES computational platform, other cloud-based design platforms for urban analysis are introduced in Section 2.

Funded by the European Union

HISTORY

Version	Date	Reason	Revised by
v0.1	24.01.2022	ToC	Verena Vogler, Ayman Moghnieh, Anne Mimet, Shany Barath, Tina Selami, Surayyn Selvan, Wolfgang Weisser
v0.2	26.01.2022	First draft	Verena Vogler, Ayman Moghnieh, Anne Mimet, Tina Selami, Surayyn Selvan, Francesca Mosca, Wolfgang Weisser
v0.3	08.03.2022	Second draft	Verena Vogler, Ayman Moghnieh, Anne Mimet, Shany Barath, Tina Selami, Surayyn Selvan, Francesca Mosca
v0.4	22.03.2022	Third draft	Verena Vogler, Ayman Moghnieh, Anne Mimet, Shany Barath, Tina Selami, Surayyn Selvan, Francesca Mosca, Luis Fraguada
v1.0	28.03.2022	Final version	Verena Vogler, Ayman Moghnieh, Anne Mimet, Shany Barath, Tina Selami, Surayyn Selvan, Francesca Mosca, Wolfgang Weisser

AUTHOR LIST

Organisation	Name	Contact information
McNeel	Verena Vogler	verena@mcneel.com
TUM	Anne Mimet	anne.mimet@tum.de
McNeel	Ayman Mognieh	ayman@mcneel.com
TECHNION	Surayyn Selvan	surayyn@campus.technion.ac.il
UNIGE	Francesca Mosca	francesca.mosca@edu.unige.it
TU Wien	Tina Selami	tina.selami@tuwien.ac.at
TECHNION	Shany Barath	barathshany@technion.ac.il
TUM	Wolfgang Weisser	wolfgang.weisser@tum.de
McNeel	Luis Fraguada	luis@mcneel.com

EXECUTIVE SUMMARY

This report addresses the overall technical requirements associated with the implementation of the envisioned ECOLOPES platform, which aims to provide innovative support for the design, valuation, and optimisation of ecological envelopes.

First, the state-of-the-art of computational platforms that address similar or related design challenges is reviewed by introducing and describing different exemplary platforms, including the Rhino framework on top of which major parts of the ECOLOPES platform are to be built and deployed (Section 2).

The report presents the design exercises conducted, primarily in WP3, to design the ECOLOPES platform and elicit the technical requirements associated with its implementation.

Then, the report discusses the design of the platform's computational framework (Section 3), which defines the software components and environments that together implement the ECOLOPES computational flow, generating, evaluating, and optimising designs. Based on the design of the computational framework, the technical requirements pertaining to the system architecture design and deployment are elicited.

Afterwards, the report describes the knowledge framework including the design and implementation of the MiMo experiment that concentrates on developing and integrating the ecological models and describing the data management and orchestration associated with their execution (Section 4). This discerns the specific requirements associated with running the models en-masse in a manner that generates meaningful design-related knowledge.

Based on these collaborative design exercises and the complementary information collection activities conducted in WP3, the data specifications related to the platform's major components as well as the overall design of the platform's system architecture are described and documented (Section 5).

Lastly, Section 6 of the report introduces the 1st prototype of the ECOLOPES computational platform – the ECOLOPES Sandbox. The section discusses the infrastructure deployed for supporting the development and integration of the ECOLOPES platform, which was conceptualised, developed, and deployed to provide an environment for data sharing, and testing computational programs and algorithms with project-related datasets. Furthermore, an example demonstrates how the frontend of the computational platform can be implemented as a Grasshopper plugin.

The report concludes by drawing plans for the next stage of the development and integration of the ECOLOPES platform.

ABBREVIATIONS AND ACRONYMS

AEC	Architecture, engineering and construction industry
AFG	Animal functional groups
AI	Artificial intelligence
API	Application programming interface
AR	Augmented reality
CAD	Computer-aided design
CFD	Computational fluid dynamics
Computational Workflow	The overall process, implemented in the ECOLOPES platform, and composed of chained software components that together resolve ECOLOPES design cases.
DEM	Digital elevation model
ECOLOPES	Ecological building envelopes
ECOLOPES Platform	A set of interoperable tools and software components that together support the ECOLOPES approach for the design of *envelopes*.
FG	Functional groups
GH	Grasshopper, a visual programming interface for Rhino
InFraReD	Intelligent framework for resilient design
KB	Knowledge base
KPI	Key performance indicators
MADM	Multi-attribute decision-making
MiMo	Mini Model
ML	Machine learning
MOO	Multi-objective optimisation
.NET	The .NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows.
NURBS	Non-uniform rational basis splines
OpenFOAM	A free open-source CFD software developed primarily by OpenCFD Ltd
PFG	Plant functional groups
Rhino	Rhinoceros, a 3D free-form NURBS modelling software and cross-platform open developer platform

Sandbox	Sandbox is a cloud-based playground for testing software components in an agile process.
SDK	Software development kit
SO	Specific objectives in the ECOLOPES project
SQL	Structured query language
UI	User interface
URL	Uniform resource locator
USLE	Universal soil loss equation
VR	Virtual reality
WP	Work package

TABLE OF CONTENTS

Author list 2

Executive summary 3

Abbreviations and acronyms 3

Table of Contents 6

Introduction 9

2. The State-of-the-Art for data-driven urban planning platforms 11

2.1 Flux.Broward.Land – an open urban development platform 11

2.2 Cityplain – a cloud computing tool for urban planning 12

2.3 InFraReD – a cloud computing tool for urban planning 13

2.4 Rhino as an open development platform and Rhino.Compute 15

2.4.1 Rhino as a cross-platform development framework for the ECOLOPES plugin 16

2.4.2 Rhino.Compute and the Rhino.Compute AppServer 16

2.4.3 The Grasshopper Hops component 18

2.4.4 Existing ECOLOPES-relevant Grasshopper plugins for urban design and analysis 18

2.5 Conclusions 19

3. The computational framework in ECOLOPES 19

3.1 Open and expert databases 22

3.2 Environmental models (WP3–WP7) 23

3.3 The Ecological Model (WP4) 23

3.3 Knowledge Base (KB) 23

3.4 The EIM ontology (WP4) 24

3.5 The design generation and optimisation environment 24

3.5.1 Architectural design components (WP5) 24

3.5.2 Analysis components (WP3–WP7) 25

3.5.3. Simulation and optimisation components (WP6) 25

3.3 Conclusions 26

4. The knowledge generation framework and the MiMo experiment 27

4.1 Goals of the MiMo experiment 27

4.2 General structure of the MiMo experiment 28

4.3 MiMo inputs 29

4.4. MiMo models 31
4.3.1 The soil depth model 32
4.3.2 The connectivity model 33
4.3.3 The solar radiation model 33
4.3.4 Water retention model 34
4.3.5 The local ecological model 34
4.4 The Knowledge base (KB) 34
4.5 Conclusions 35
5. Software development approach for ECOLOPES 35
5.1 Data and process specification for all components 35
5.2 Conclusions 37
6. The ECOLOPES system architecture 37
6.1 The software components 40
6.1.1 The EIM ontology component 40
6.1.2 The ECOLOPES algorithms component 40
6.1.3 The ECOLOPES data warehouse component 40
6.1.4 The ECOLOPES computational simulation environment component 41
6.1.5 The ECOLOPES front-end tools component 41
6.1.6 The ECOLOPES Multi-Species Habitat component 41
6.2 Advantages of the drafted system for the ECOLOPES project 41
6.3 Technical requirements for building the ECOLOPES platform 41
6.4 Conclusions 42
7. The sandbox – a cloud-based platform for ECOLOPES 43
7.1 The sandbox – the 1st prototype of the computational platform 43
7.2.1 Data storage 44
7.2.2 Rhino.Compute server 45
7.2.3 The algorithm production server 46
7.2 Technical details on the sandbox setup 46
7.3 Testing of the sandbox: ECOLOPES plugin for Grasshopper 47
8. Conclusions and recommendations for the next version 48
References 50
Research Paper: 50
Technologies: 50

Software libraries and tools: 51
Appendix 52

1. INTRODUCTION

The ECOLOPES research project envisions a radically new integrated ecosystem approach to architecture, including the development of a new technology to achieve this vision. The new technology aims at enabling a design system for ECOLOPES – a new building envelope that includes the requirements of multi-species inhabitants. Such a holistic approach to species will allow plants, microbiota, animals, and humans to co-evolve within future cities (D4.1). However, the main challenge for the development of a joint technology is a common understanding between the multidisciplinary experts involved, mainly between architecture and ecology.

This includes the exchange of knowledge concerning methods, workflows, datasets, sub-techniques, data exchange formats, with the goal to further implement the key aspects of the ECOLOPES technology:

1. to make **ecological knowledge** available for architectural design; and
2. to develop a **simulation environment** that generates architectural design outcomes that enable synergies and, thus, limit conflicts between the multi-species inhabitants (Multi-species design optimization within the CAD environment).

The D3.1 report reflects on the complexity of such an undertaking and focuses on the individual steps towards the ECOLOPES technology. It describes the progress achieved in the conceptualization, design, and implementation of the ECOLOPES Platform, which will convey the ECOLOPES approach and associated computational and analytical processes to the designers and architects targeted as users.

In particular, the report details the high-level technical requirements of the ECOLOPES Platform, including the workflow that needs to be implemented, data specifications for different components, system architecture and enabling infrastructure and middleware components, backend integration, and the functionalities of the envisioned tools.

During the first phase of the project, and in order to design the ECOLOPES system architecture and define its constituent components, a benchmarking of related digital technologies and design frameworks was conducted, focusing on cloud-enabled systems and urban design applications that support functionalities similar to those contemplated in ECOLOPES. A selection of these applications and technologies is presented in section 2, **state-of-the-art of relevant applications**, focusing on selected platforms and technologies that enable a more analytical, data-driven, and holistic approach to urban design and urban planning, encapsulating some of the concepts and processes of ECOLOPES.

The overall process implemented and supported by the ECOLOPES Platform is referred to as the computational workflow, and its design is crucial for defining its architecture. A consolidated design of the computational workflow was achieved, as a fruit of collaborative work between all technical partners involved. It encapsulates the overall technical requirements of the ECOLOPES platform and shows how the different computational components are connected and how data passes through the system. The current **design of the computational workflow** that ought to be supported by the ECOLOPES platform is presented in section 3.

In order to analyse the technical requirements of the ecological model and to establish the data management processes associated with the generation of basic knowledge for the ECOLOPES platform, the ***MiMo experiment*** is introduced and discussed in section 4. It showcases how the different elemental ecological models are integrated into a composite ecological model, which can be executed to create datasets pertaining to the generation of knowledge.

Based on the outcomes of the collaborative design exercises conducted to conceptualise and describe the computational workflow and the MiMo experiment, the preliminary **data specifications** that define the input/output format and requirements for all components are elicited. These specifications are described in section 5, including aspects pertaining to all major components of the ECOLOPES platform.

The technical requirements from a system architecture standpoint describe the necessary infrastructure for the ECOLOPES platform and its components, as well as its integration and deployment models. The technical requirements specify how to support data management, backend functions, and user processes. This overall **system architecture** is described in section 6, reflecting on the selected architecture model's advantages and capabilities.

Based on the conceptual design of the system architecture, a digital infrastructure and a development and deployment environment has been created and deployed in order to support the activities related to the functional bottom-up building of the ECOLOPES' service architecture, in particular enabling the deployment of advanced computational algorithms for design generation, analysis, optimisation and evaluation. This architecture is called the **Sandbox**, and is a prototypical version of the ECOLOPES platform. It is a cloud-based environment where agile processes can already be deployed and tested. It uses cloud-based computing, data storage, and data sharing options with the goal to provide a joint environment for all collaborators. The ECOLOPES Sandbox is described in section 7.

Concerning the research methods related to the elicitation of these technical designs and technical requirements, in the first year of the project mainly quantitative strategies (data specification questionnaires, interdisciplinary workflow discussions) were applied. However, lately, empirical methods as part of an experimental exploration process have become more relevant for the development of the first prototype of the ECOLOPES platform. A particular example is the MiMo experiment, a computational experiment that aims at building up knowledge about the relationships between architectural and ecological aspects (Section 4).

Our preliminary results are a first version of the computational workflow, data specification for all computational components, the design of the **MiMo experiment** and expected outcome including the definition of obstacles, successful implementation of a cloud-based infrastructure through the Sandbox, and finally, the definition of the technical requirements for the ECOLOPES front-end tools.

The knowledge acquired during the conduction of the activities described, and the overall technical specification and planning drafted, will enable and support the agile integration and deployment of the first version of the ECOLOPES platform in the coming project period. The technical foundations achieved so far constitute a clear roadmap for the integration of the different components developed in the project work packages, and demonstrate the technical viability of the envisioned ECOLOPES platform.

2. THE STATE-OF-THE-ART FOR DATA-DRIVEN URBAN PLANNING PLATFORMS

One of the main challenges with respect to the ECOLOPES technology is to combine cross-disciplinary expert datasets with 3D CAD models, and to enable a user interface to interact with the system. In urban design research and software development practice, similar challenges were encountered and addressed.

As the digital transformation of design processes deepened, more data-driven and data-intensive processes have been incorporated in general practice and consequently in the supporting software platforms. Advanced enabling technologies (e.g. sensing and sensorisation, AI-driven data modelling and simulation, machine learning and forecasting, etc.) that generate and leverage this data require higher computational power and a more decentralised approach to system architecture. Therefore, computational frameworks and software for architectural design have been migrating partially or completely to cloud-based scalable architectures. Urban planning and analysis software solutions are no exception.

In this section, prominent and relevant examples of contemporary cloud-based urban design platforms are reviewed: Flux.Land – an open urban development platform (Section 2.1); CITYPLAIN, a cloud computing tool for urban planning (Section 2.2); and InFraReD – a platform for intelligent and resilient urban design based on Artificial Intelligence (Section 2.3). Finally, Section 2.4 offers a review of the Rhino.Compute platform with respect to its relevance for the development of the ECOLOPES design platform. In this context, it introduces another example of a web-based interface for urban analysis built on top of Rhino.Compute, the KPF Urban Interface.

2.1 Flux.Broward.Land – an open urban development platform

Flux.Broward.Land, developed by the Urban Risk Lab at Massachusetts Institute of Technology and the Center for Landscape Research at the University of Toronto, is an interactive, web-based, geospatial platform designed to increase awareness and bridge the gap between different stakeholders of urban development. It offers a data-driven, collaborative, web-based toolkit for urban planning and decision-making across the scales (Seah et al, 2021). The front-end of Flux.Broward.Land can be accessed through a standard web browser (Flux.Broward.Land, 2021). Methods for the integration of data visualisation, data analytics, and data-driven design have been developed for the platform. Furthermore, a sectional tool correlates elevational information of streets, water bodies, and future sea-level rise impacts on groundwater tables (Figure 1).

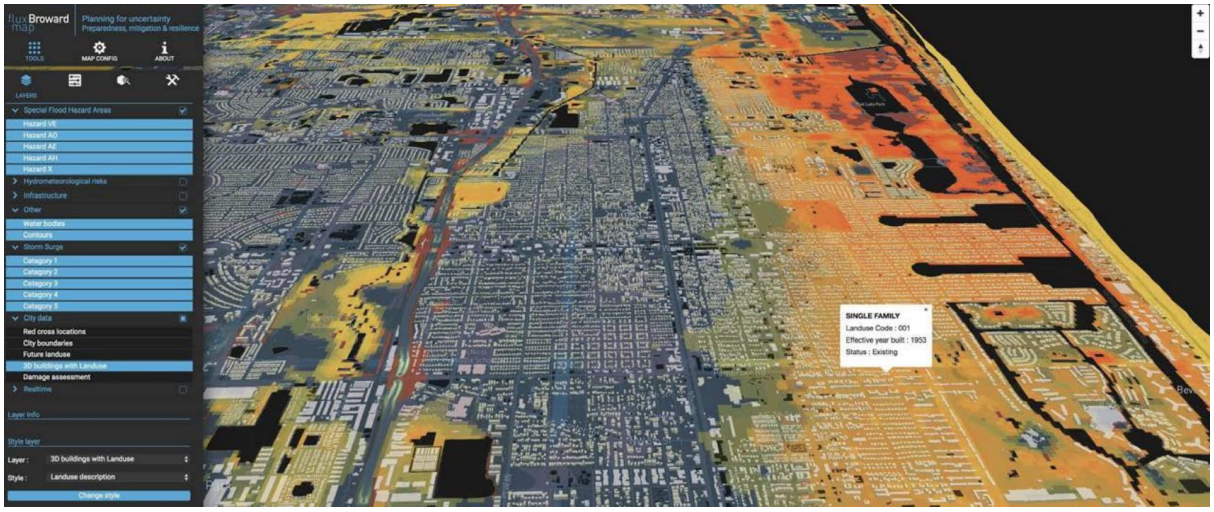

Figure 1: Flux.Broward.Land's user interface in a standard web browser (Flux.Broward.Land, 2021).

In summary, the platform is a great example of a user-friendly application for cloud-based urban planning and analysis. The emphasis was clearly placed on the platform's simple navigation and design aimed at stakeholders from varying degrees of expertise – from the general public to planners. However, there were major challenges with respect to hierarchical data clustering to elicit underlying similarities and differences within areas of interest beyond simple visualisation of datasets across administrative boundaries.

Using contextual-specific data points and clusters generated on the platform, a practitioner can leverage an information database of the platform to make better design decisions. The integrated data-driven design methodology can be summarised in three key steps: (1) defining design priorities and goals, (2) quantifying evaluation metrics and design parameters, (3) design discovery, optimisation, and curation of design recommendations.

Through the Flux.Broward.Land example, it becomes clear that the development of a cloud-based interactive platform comes along with many challenges such as data clustering, the establishment of an information database (knowledge base), and the visualisation of the 3D design and analysis outcome for stakeholders of diverse backgrounds. However, Flux.Broward.Land remains a design-recommendation tool based on analysis. In the next example, the focus is on a cloud-based platform that enables urban design and planning (Section 2.2).

2.2 Cityplain – a cloud computing tool for urban planning

Cityplain is a cloud-based platform for urban planning that addresses affordable housing within city extensions developed by Citythinking, an urban consultancy specialised in the management and design of integrated urban solutions based in Sevilla. Cityplain allows comprehensive scenario comparisons and cross-team collaborations to improve the design decision-making processes. As a nature-based and data-driven solution, it aims at making more sustainable design decisions for residential districts. The platform uses functional, environmental and socio-economic key performance indicators (KPIs). These are optimised in relation to the geometry of a design stage and also for Cityplain to learn from stage to stage design patterns to suggest more precise and efficient design solutions. Thus, they are integrated within an AI-based design recommendation system which is still under

development. Furthermore, the tool combines 3D geometry and numerical dataset from the analysis. As a web-based platform, it can be accessed by team members and stakeholders, allowing everyone to be part of the decision-making process (Cityplain, 2021) (Figure 2).

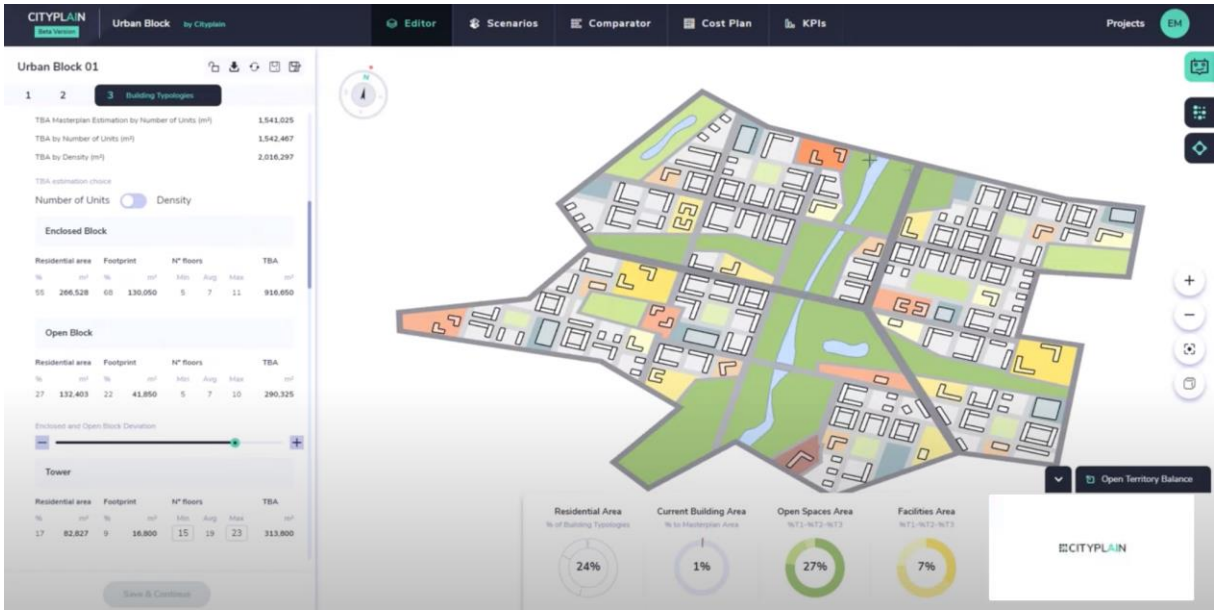

Figure 2: The web interface of Cityplain (Cityplain, 2021).

In summary, Cityplain is a web tool to inform stakeholders to make more efficient and sustainable design decisions for residential districts in urban areas. It provides analysis results from an evaluation process using KPIs and in the future version also by an AI-based ranking system of KPIs. However, the tool is limited in terms of accessibility. It is a commercial software solution and, thus, restricted to a limited user group. The high number of parameter inputs makes for a complex interface, and not all users might be familiar with defining parameters that require expert knowledge. Finally, the tool does not provide a convincing solution for designing urban building infrastructure considering terrain modelling. However, a GIS integration is foreseen for the next version. Currently, the design space is still on a plane without information about the elevation of the terrain, hydrological, geomorphological, and biological applications, which might be sufficient in the context of Cityplain, but such information would be relevant for a sustainable development of urban environments and buildings in ECOLOPES (Moore et al, 1991).

2.3 InFraReD – a cloud computing tool for urban planning

The Intelligent Framework for Resilient Design (InFraReD) was developed by many institutions including the Austrian Institute of Technology, the City Intelligence Lab, and the Bauhaus-University Weimar (InFraReD, 2021). It is a platform for intelligent and resilient urban design based on Artificial Intelligence (AI). Machine learning (ML) models provide real-time feedback on the performance of custom designs and guide decisions at every step of the process. The performance is measured by Performance Indicators (Figure 3). InFraReD currently provides feedback on solar, sunlight and wind performance. Additional analysis methods will be included soon (Figure 4).

Figure 3: The web interface of InFraReD computes performance indicators for urban analysis (InFraReD tutorial, 2021).

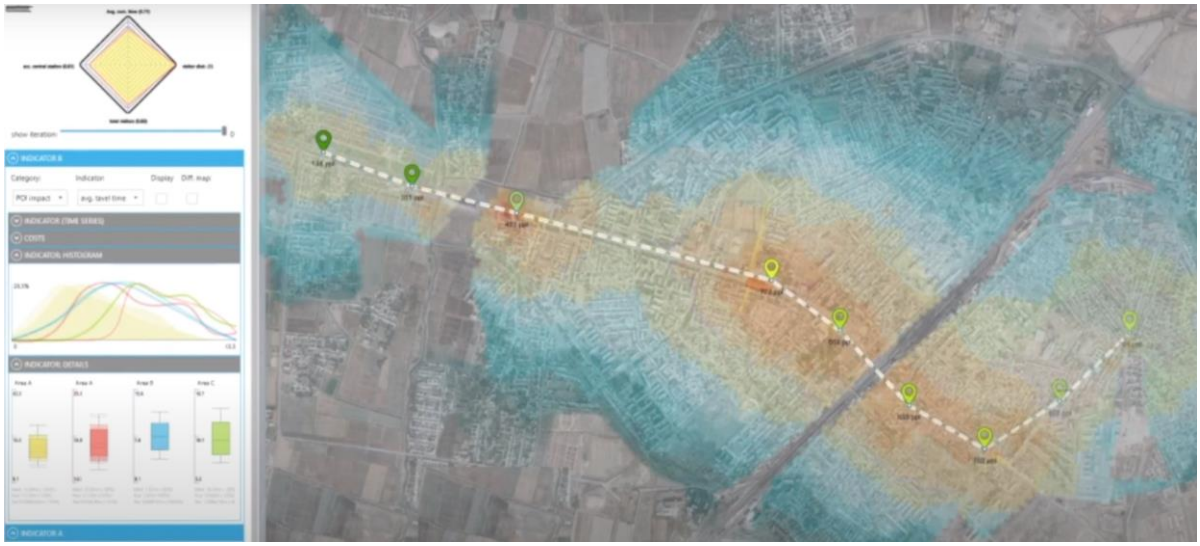

Figure 4: The web interface of InFraReD computes the correlations between performance indicators (bottom, left) and applies the analysis results as a heatmap on the urban area (InFraReD tutorial, 2021).

InFraReD fuses AI, Augmented Reality (AR), and computational design into one system. Thus, it is not only digital but also a physical and intellectual space for urban planners (Figure 5). Urban analysis inference models are trained by applications that are parts of the Rhino ecosystem, such as Grasshopper, Ladybug Tools (Ladybug Tools, 2013), and other custom-developed tools by the InFraReD development team (Duering et al, 2020). This is then passed to OpenFOAM, a free open source CFD software developed primarily by OpenCFD Ltd, for generating the CFD results (OpenFOAM, 2004).

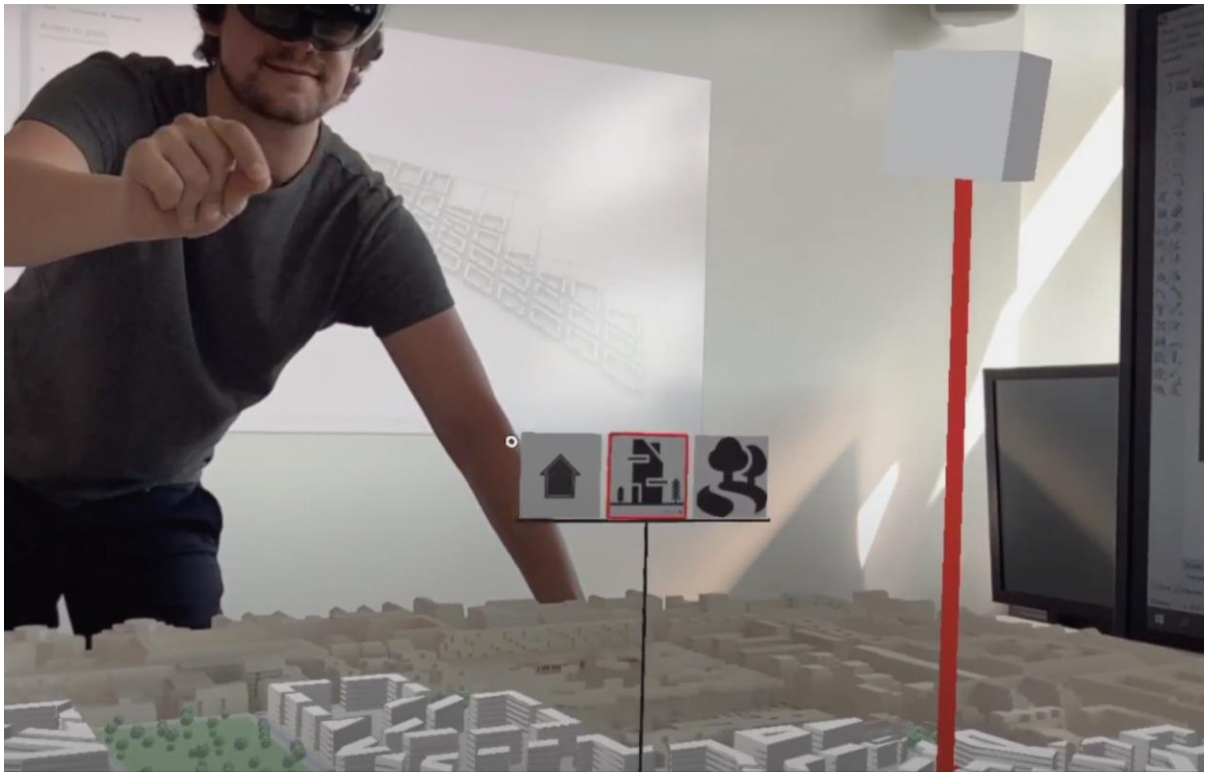

Figure 5: Augmented reality interface as part of the InFraReD technology (InFraReD tutorial, 2022).

In conclusion, InFraReD is a cutting edge cloud technology for AI-based urban analysis in AEC. Besides an interactive web-based user interface, it also provides a physical AR interface for decision-makers. AI includes trained correlations between the most relevant performance indicators for urban planners such as wind, solar radiation and sunlight with the possibility to visualise analysis results in real-time for the user of the platform.

2.4 Rhino as an open development platform and Rhino.Compute

Rhino is a 3D free form modelling application developed by Robert McNeel & Associates, a privately-held, employee-owned software development company based in Seattle. 3rd party developers can extend Rhino's functionality through freely available SDKs and there are more than 800 plugins available to date. Rhino has become increasingly popular in the AEC industry when Grasshopper, a visual algorithmic editor developed by David Rutten, became part of the software in 2008 (Grasshopper, 2008). Nowadays, there are more than 600,000 active Rhino licences and the user community in the McNeel forum has about 80,000 active users. Most stakeholders are from the AEC industry and industrial design, but also footwear, jewellery, and marine design are gaining in relevance. Rhino 7, the most significant Rhino version in the development history of McNeel was released at the beginning of 2021 (Rhino 7, 2022). Its success can be attributed to the inclusion of the new free-form modelling method based on SubDivision, but also due to its possibilities to run inside other 64-bit Windows applications (Rhino.Inside, 2021), e.g., the BIM modelling software Revit developed by Autodesk (Rhino.Inside.Revit, 2021), and 'headless' as a cloud computing service (Rhino.Compute Guides, 2021). Thus, Rhino 7 unlocked completely new modelling and development workflows.

The following sections introduce Rhino's open development framework (Section 2.4.1), Rhino.Compute and the Rhino.Compute AppServer as a way to cloud-compute 3D geometry, and to visualise it through a web-interface (Section 2.4.2). Section 2.4.3 presents the capabilities of cloud-computing for parametric models using the Grasshopper Hops component, and finally, additional Rhino and Grasshopper applications that are crucial for enabling the computational workflow in ECOLOPES are discussed.

2.4.1 Rhino as a cross-platform development framework for the ECOLOPES plugin

The ECOLOPES platform interfaces with the user through the **ECOLOPES front-end tools** (Task 3.4, frontend development), which will be developed as a **plugin on top of Rhino**. Rhino is an open development platform (Rhino Development, 2018). Thus, it provides a Software Development Kit (SDK) as well as guides that can be applied across platforms – Windows and OS X.

Plugin development for Rhino for Windows: The Rhino SDK is a set of royalty-free developer resources for customising and extending Rhino for Windows. It provides documentation, tutorials and tools, as well as software libraries. The Rhino C/C++SDK consists primarily of C++ headers and libraries that can be used to build Rhino extensions called Plugins. Plugins are Windows DLLs that can be loaded into the Rhino process and interact directly with the Rhino application (Rhino SDK, 2018).

Plugin development for Windows and for OS X: For the development of cross-platform Rhino plugins, Rhino provides a .NET SDK called RhinoCommon to create compiled code libraries. With this .NET SDK, a developer can create plugins for Rhino typically in C# or VB.NET. These plugins can be as simple as adding a new command to Rhino, all the way to extensive systems which introduce new UI elements and object types. Another SDK is included to create add-ons for Grasshopper, which help in developing custom component libraries.

Grasshopper (GH) is a RhinoCommon (.NET) plugin for Rhino 7 for Windows and Mac. It was written using Microsoft Visual Studio Professional using both VB.NET and C# source compiled against the .NET Framework (Grasshopper component, 2018).

Another possibility to develop cross-platform plugins is Rhino IronPython (Iron Python, 2009) which brings together the Python language and Microsoft's .NET framework (Rhino.Python, 2018). For Rhino plugin development, IronPython is used, for instance, to perform tasks in Rhino or Grasshopper, and to generate geometry using algorithms.

With respect to the **ECOLOPES plugin,** it means, that custom algorithms can be written either as components for Grasshopper or a plugin for Rhino and these can be written in C#, C++, or Python language.

2.4.2 Rhino.Compute and the Rhino.Compute AppServer

Rhino.Compute is an open-source project. It uses the Rhino.Inside™ technology (GitHub Rhino.Inside, 2021), another open source project which allows Rhino and Grasshopper to run inside other 64-bit Windows applications. In Rhino.Compute, Rhino SDK functions are accessed via a cloud-based stateless REST API. Through Rhino.Compute developers can create applications that manipulate Rhino (3DM files), solve Grasshopper definitions and python scripts without needing to have Rhino installed on the client devices. Furthermore, it can call

over 2400 geometric operations from the RhinoCommon SDK remotely. With respect to the ECOLOPES platform, Rhino.Compute offers the possibility to calculate process-intensive Grasshopper algorithms for geometry generation and analysis within the cloud. Furthermore, it allows cross-disciplinary team members to have access to a 'headless' Rhino version and to work as a team simultaneously on the same project. Another advantage of Rhino.Compute is that Grasshopper definitions can run as a **backend service**. In this context, the Rhino.Compute AppServer is a node.js server acting as a bridge between client apps and private Rhino.Compute servers. The app is intended to host one or more custom grasshopper definitions and serve as the API that client applications can call to have definitions solved with modified input parameters (Rhino.Compute AppServer, 2021). Thus, the Rhino.Compute AppServer allows displaying parametric models in a web browser.

Rhino.Compute and the Rhino.Compute AppServer have become relevant for the development of cloud-based design and analysis platforms in AEC. For instance, the international architecture and engineering practice, Kohn Pedersen Fox (KPF), developed together with other research institutions the KPF Urban Interface (KPF UI, 2022). It is an open interface that uses urban data analytics for an informed and more efficient decision-making process for building design and the planning of more sustainable cities. The KPF UI uses methods for real-time visualising urban data analysis, and displays the data in a way that it is understandable for users (Scout.Build, 2021). Furthermore, real-time analysis enables to match stakeholder assumptions with data analytics (Figure 6). Another relevant example in this context is Thornton Thomasetti's Swarm app (Swarm, 2021) It uses Rhino.Compute and the AppServer to connect designers in design teams for the design exploration and decision-making process in a web browser.

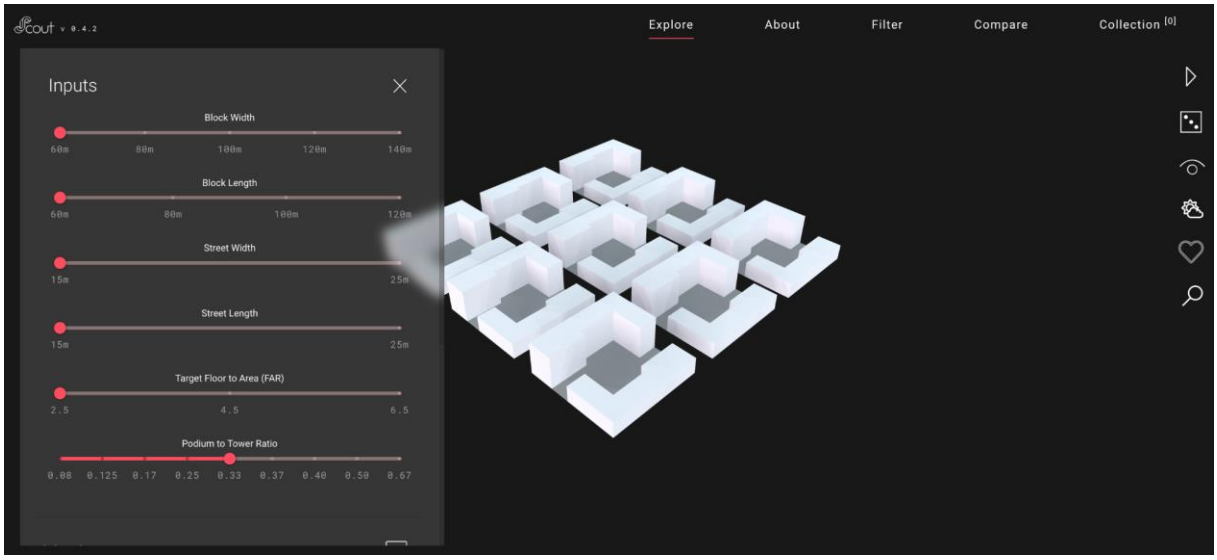

Figure 6: Computational design model in KPF's Scout web platform (Scout.build example, 2021).

In summary, using the Rhino.Compute AppServer in ECOLOPES would enable a hardware and software independent visualisation of parametric models in a standard web browser. Also, non-expert users could easily interface with the ECOLOPES design platform through this application. Furthermore, the developed Grasshopper algorithms for geometry generation,

analysis and simulation could be used to run as a back-end service and results could be displayed in real-time on the web or on a local machine.

2.4.3 The Grasshopper Hops component

Hops is a Grasshopper component in Rhino 7 (Hops, 2021). It can compute Grasshopper definitions that are stored on a cloud or on a local server via file path or URL. Thus, it can call Grasshopper definitions from an online algorithm library and solve them locally. Such a process is not only useful to simplify complex algorithms, but also to share the same algorithm with team members, and referencing it across multiple projects. All Grasshopper plugins can run within a Hops function (Figure 7).

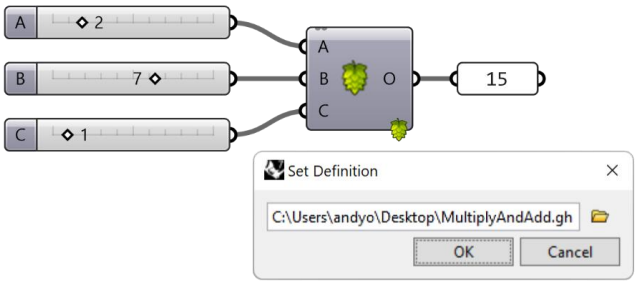

Figure 7: The Grasshopper Hops component solves another Grasshopper function without having to open it (Hops, 2021).

By default, Hops uses a local computer to solve Grasshopper functions. However, it is also possible to set up remote servers or virtual machines for Hops to call. Furthermore, Hops adds external functions to Grasshopper, e.g. other programming languages and functions. This way, Hops can call into CPython libraries including Numpy, SciPy, or TensorFlow. It can also work alongside Rhino.Inside.CPython to give full access to the Rhino Common API.

In respect to the ECOLOPES platform, Hops is a useful component to cloud store and share the ECOLOPES algorithms developed by other work packages (WP). Further, Hops can interface with TensorFlow to introduce machine learning for training algorithms and models in ECOLOPES.

2.4.4 Existing ECOLOPES-relevant Grasshopper plugins for urban design and analysis

To get a better understanding of the variety of algorithms that can be developed within Grasshopper, the following section provides an overview and discussion about a pre-selection of additional plugins:

Ladybug Tools for solar radiation analysis and for daylight optimisation: Ladybug Tools is a collection of free computer applications that support environmental design in Grasshopper (Ladybug Tools, 2013). It is built on top of validated simulation engines such as Radiance, EnergyPlus/ OpenStudio, Therm/ Window and OpenFOAM. Ladybug Tools is written in Python, which can be run on virtually any operating system and plugged into any geometry engine. The project is open-source so Ladybug Tools Grasshopper components can be edited

and rewritten. For ECOLOPES, Ladybug Tools could be an important tool to compute solar radiation (Section 4.4), and for computing human comfort (Section 3.5.2).

Decoding Spaces for urban design and analysis: Decoding Spaces is a free computational analysis and design generation tool for street networks, plots, and buildings developed at the Bauhaus-University Weimar (DeCodingSpaces Toolbox, 2017). As a Grasshopper plugin, its algorithm library contains urban analysis methods for data analysis which can be combined with stakeholder requirements (KPIs) to make more efficient design decisions in ECOLOPES.

Wallacei for evolutionary computation: Wallacei is a free evolutionary multi-objective optimisation and analytic engine for Grasshopper that allows evolutionary computation in design (Wallacei, 2020). As an open-source project, Wallacei can be extended. In ECOLOPES, Wallcei could play a crucial role, as it already provides evolutionary computation algorithms for generative design (WP5 and WP6).

Docofossor for terrain modelling: Docofossor is a free parametric tool in Grasshopper to transform digital elevation models (DEM) by point, path, area, or surface. In contrast to surface models, DEM are required for water retention modelling, land-use studies, or other geological applications. The plugin is written in IronPython and can be extended. It was developed at the Building Technologies department at ETH Zurich (Hurkxkens & Bernhard, 2019), (Docofossor, 2019).

2.5 Conclusions

The presented platforms, solutions, and technologies showcase the advances of real-time cloud computing, visualisation of results in open web interfaces, and the implementation of decision-support tools for urban design and urban analysis. They serve as a reference for the design and development of the ECOLOPES platform that aims to build on top of compatible frameworks and system architecture paradigms. Technically, the technical requirements fulfilled by these state-of-the-art platforms, in particular, their abilities to optimise highly demanding processes and to provide on-the-fly data analysis capabilities, are used as an initial reference to define the functional and non-functional requirements that ECOLOPES platform aims to support.

The review in this section has demonstrated that Rhino's open framework provides an extendable, and customisable environment for the development of such platforms, allowing developers to deploy expert programs as a service, and chain programs from different providers addressing distinct concerns of analysis, evaluation, simulation, and generation of design components. For this purpose, it will be exploited in the development and integration of the ECOLOPES platform.

3. THE COMPUTATIONAL FRAMEWORK IN ECOLOPES

WP3's main responsibility is the creation of the **ECOLOPES computational platform (SO1)**. In order to design the architecture of the ECOLOPES platform, the data and computational flows were examined in detail collectively with the developers of the different components and constituents. In this exercise, the technical and non-technical requirements were taken into account in order to fulfil the project's technical objectives. In addition, a special focus was

dedicated to defining the data objects that are created, exchanged, and analysed in this workflow, including Raster data, tabular data, dynamic data, static/referential data, voxel models, CAD models, among others. The computational components developed in the other work-packages were initially addressed as 'black boxes' to emphasise their input/output requirements and their technical requirements in terms of middleware, computational resources, and other concerns.

The results of this technical analysis applied to the envisioned workflow are encapsulated in the computational framework. It describes the data-driven components and processes of the ECOLOPES platform and provides a data governance framework and a high-level system architecture design that defines and delimits the role of technical components supporting the ECOLOPES design and analysis processes. It also explains how envelopes are technically composed, and the roles and contributions of different components.

The computational framework essentially rearranges the assets of the ECOLOPES platform and consolidates them into five major modules, each responding to specific data and functional requirements. The modules are: the open and expert databases, the ecological model, the knowledge base, the ontology, and the design generation and optimisation environment (Figure 7).

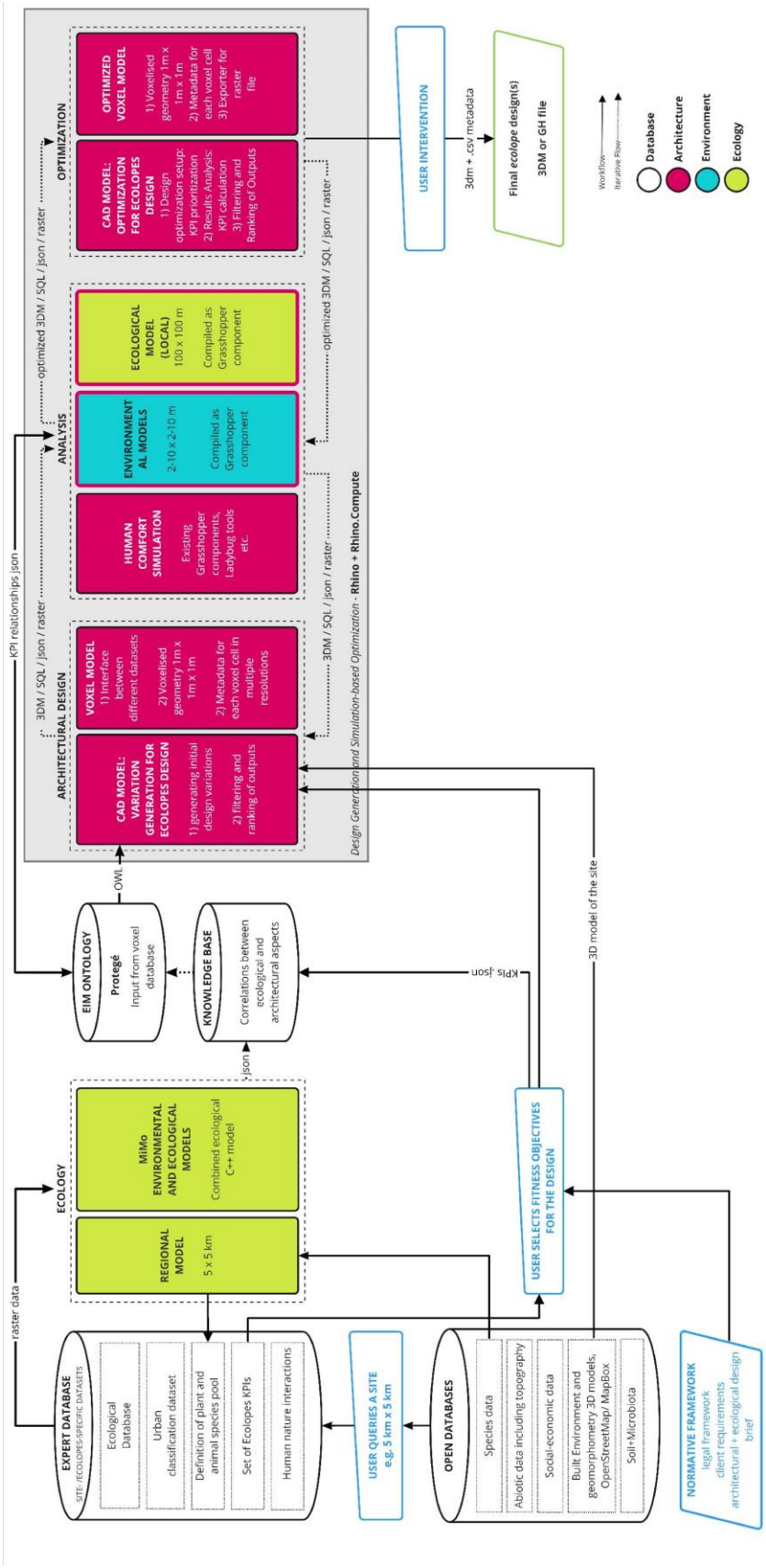

Figure 7: The computational framework for the development of the ECOLOPES platform.

From a workflow perspective, the system connects external data sources and also creates and hosts expert datasets, which are then used as referential data on aspects such as species, ecosystems, climates, geography, urbanism, and built environment. The ecological model simulates an ecology under specific conditions and can be run iteratively and continuously to identify correlations among conditional variables, in particular variables with geometric (design) significance. This generated elemental knowledge pertains to the system ontology, which identifies and describes design-relevant relations and different variables of the *ecolope*. The ontology builds such relations on top of this generated knowledge (in addition to other expert knowledge), and is consulted by the design, analysis, and simulation programs. These aforementioned components constitute the data and knowledge management part of the ECOLOPES platform. Together, they aim to capture, structure, generate, interrelate, and serve the fundamental data concerns that are exploited by the ECOLOPES services and tools in an ad-hoc manner.

The following sections introduce the five individual modules of the computational framework in more detail and explain how they are interlinked.

3.1 Open and expert databases

Open databases are publicly available data sources on species, soil, abiotic concerns, built environment, available local 3D assets, and other concerns, which are pertinent to the composition of the expert databases and occasionally to the execution of the ecological models. In contrast, the expert database, it contains datasets that have been compiled from open sources and expert models to capture computationally-relevant concerns (e.g. species pools, KPIs, etc.), or to describe ECOLOPES-related concerns, such as human-nature interactions (Figure 8). The expert database is regularly queried by ecological models.

While open databases are already available through 3rd-parties, the expert database can be implemented as an integral module of the ECOLOPES platform, using a data storage infrastructure deployed in the cloud in order to facilitate and govern data access and use. In all cases, this data is considered largely static from a design perspective and will be managed as such.

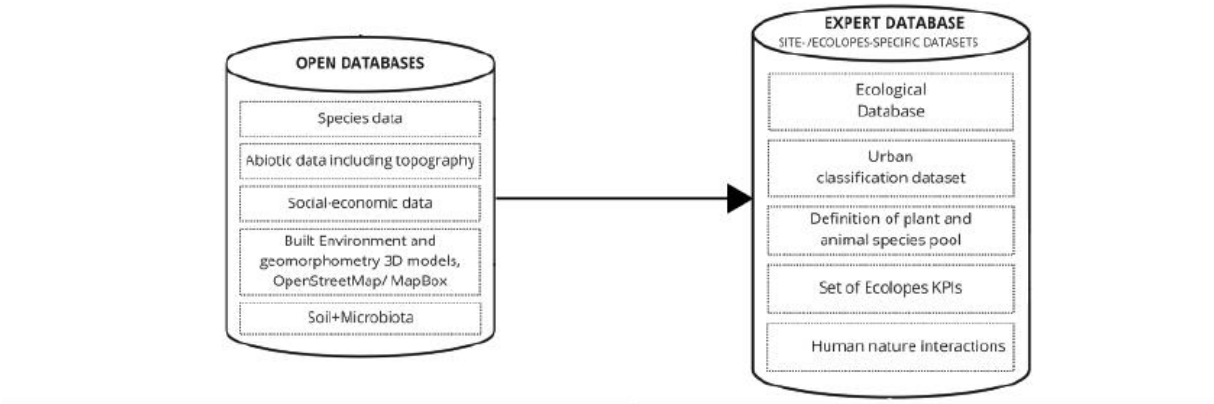

Figure 8: The open and expert databases of the ECOLOPES platform.

3.2 Environmental models (WP3–WP7)

Those models establish the connection between the architecture and the ecological model, providing necessary data (such as incoming radiation, soil depth). They are currently developed in the MiMo experiment and are further described in Section 4.4 of this deliverable.

3.3 The Ecological Model (WP4)

The ECOLOPES model is a composite spatial-explicit model that models the interdependent spatial and temporal dynamics of the soil, microbiota, plants, and animals, in response to the regional species pool, the geometry of the building, the local abiotic conditions, the substrate used to design the *ecolope*, and the management. The biological units of the model are plant (PFG) and animal (AFG) functional groups, i.e. groups of species sharing similar characteristics in the way they respond to and influence their environment. The model is based on a multi-scalar approach. The regional ecological model determines which FGs of the species pool have a reasonable chance to colonise the *ecolope* according to its location in the city. The local ecological model applies a second filter on these species based on the abiotic and biotic conditions delivered by the *ecolope*.

Essentially, the ecological model integrates all elemental models developed to address concerns related to ecology. These individual models are tightly integrated into a composite model because they are largely interdependent. Their interfaces also have been standardised by definition, so that they support the same data model in terms of input and output. This composite model can be deployed on an independent server located at, and operated by TUM, as part of its computing cluster. It can interface with other components of the ECOLOPES platform through secure HTTP requests. In addition, stable versions of the model can be installed on a cloud server for more stable deployment.

3.3 Knowledge Base (KB)

The **Knowledge Base (KB)** is a data storage system used to store structured and unstructured data resulting from the execution of the Ecological Models and the selected KPIs. It is designed to support the discovery and valuation of correlations between different data variables, namely between architecture-related variables and ecologically-related variables. Its role is to cumulate and statistically analyse the output of the ecological models at each execution, including the resulting environmental and ecological characteristics of the modelled *ecolope* (e.g. radiation input, soil depth, water retention, composition and location of different plants and animal functional groups, etc.). The knowledge base can be queried to provide statistical correlations on-the-fly that befit particular criteria (e.g. location, design parameters, climate parameters, ecological conditions, etc.). The knowledge base will be primarily populated by the results of the **MiMo experiment**, which will aim to run simulations en-masse to cover variations along with general, generic, and popular variables.

Technically, the knowledge base can be implemented as a cloud-based SQL database with a web-enabled interface, which allows other remote systems to execute queries and retrieve statistical results. It will implement the **ECOLOPES Information Model** (EIM) that defines the input-output parameters of the Ecological Models, which integrate all the computationally-significant data fields related to architecture as well as ecology.

3.4 The EIM ontology (WP4)

The EIM ontology is the reasoning framework for the ECOLOPES platform. It will leverage information from the KB and will capture existing patterns that have been proposed to define an annotation model that can be queried by other components of the computational framework. The design principles of the EIM ontology are described in Deliverable 4.1. The EIM Ontology will interface with expert data and the design generation and optimisation environment through an SQL database. For the EIM ontology, the **Protégé framework** is envisioned. This allows for flexibility in integrating the ontology with other databases in the system and developing interfacing capabilities for other ECOLOPES services to consult the Ontology and retrieve specific results.

3.5 The design generation and optimisation environment

The design generation and optimisation environment is a CAD environment built on top of Rhino / Rhino.Compute technologies (Section 2)(Figure 9). All algorithms for design generation, environmental and ecological analysis, and design optimisation are supported by the system and can be deployed as part of the ECOLOPES platform. Components that export data or save calculations on the platform's data storage system will be hosted in the cloud (at least, their backend services) in order to support more efficient security and data transaction management. In the following section, the components of the design generation and optimisation are explained in more detail: (1) Architectural design; (2) analysis; and (3) optimisation.

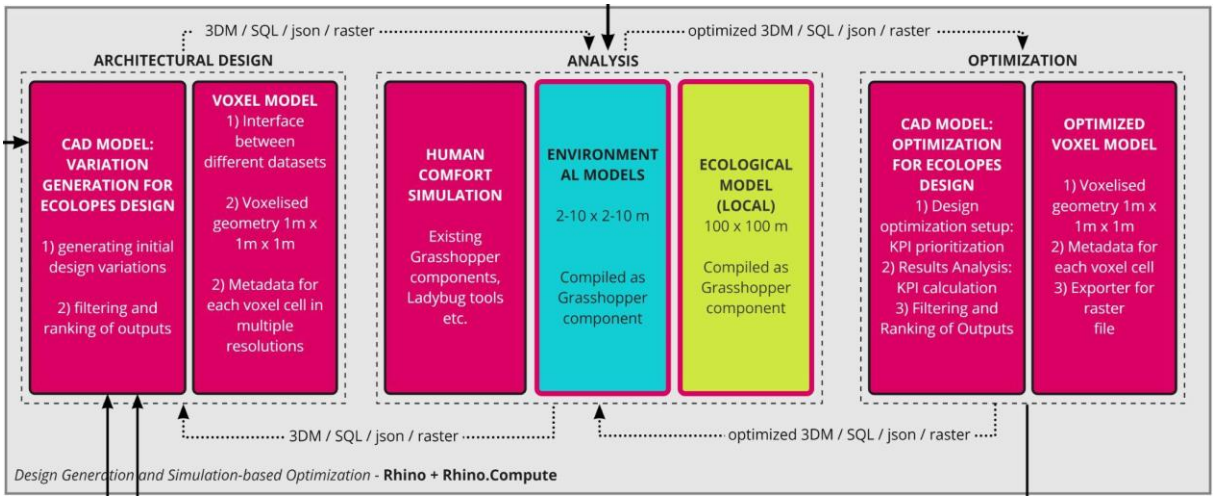

Figure 9: The design generation and optimisation environment.

3.5.1 Architectural design components (WP5)

The development of a generative algorithmic process for designing an Ecolope is conducted in WP5. WP5 has three objectives: 1) development of a Voxel model that links the EIM Ontology from WP4 with the computational model (CAD model); 2) development and integration of a generative algorithmic process within the Rhino and Rhino.Compute framework implemented by McNeel; 3) validation of the algorithmic process that delivers the

basis for the work in WP6 and WP7. Objective 2 entails the generation of design variations, from which a selection will be optimised in WP6. The algorithms for 1) generating initial design variation and 2) filtering and ranking the outputs will be connected to the CAD model. D5.1 describes the development process of the ECOLOPES algorithm. This includes, 1) the identification of relevant data sets (terrain, maps, networks, volumes); 2) identification of an approach to a generative algorithmic design process; 3) identification of principal approaches to linking the algorithmic process to the ECOLOPES voxel model and EIM Ontology. The description of the key datasets relates to the first part of the algorithmic process up to the detailed design process. The Voxel model operates as an interface between different datasets that can incorporate expert information. The Voxel data will be written in an SQL database in different resolutions.

3.5.2 Analysis components (WP3–WP7)

The analysis components are a series of existing and new Grasshopper components which will become part of the ECOLOPES plugin (front-end tool). They contain simulation modules that address environmental analysis; as well as human comfort. However, the main simulation will be conducted by the Ecological Model which computes the spatial dynamics of multi-species for one specific design outcome.

Human comfort simulation (Thermal Comfort) (WP6 and WP7): In order to evaluate thermal conditions and their impact both on the environment (in particular on buildings and their "living" components) Ladybug Tools and Morpho components will be used to run simulation models potentially comparable with the simulations that will be conducted in the validation process (WP7).

Environmental Model simulation components (WP3–WP7): These new Grasshopper components developed by WP3-WP7 will compute the a) connectivity for animals to move around, b) soil placement, c) solar radiation, and d) water retention based on one envelope design. These models are more extensively detailed in the current deliverable, section 4.4. The analysis results will be visualised and results provided for optimising the design outcome.

Ecological Model simulation component (WP3 and WP4): This new Grasshopper component allows running the Ecological model within Rhino's CAD environment. It will return simulation results in respect to the ecosystem for various design solutions. It is worth noting that the encapsulation of these analytical components as Grasshopper programs (especially the Ecological Model) will help to optimise the platform during run time as service-oriented architecture, and instantiate and replicate the platform's backend to scale or to respond to specific custom applications or use cases.

3.5.3. Simulation and optimisation components (WP6)

The optimisation process of ECOLOPES design outcomes is also a generative design process. Thus, it is similar to the design generation process in WP5, as it relies on evolutionary computation algorithms. For the optimization of the ECOLOPES design, there are the following three steps: 1) Design optimization setup; 2) results from analysis; and 3) filtering and ranking of outputs.

1. **Design optimisation setup:** The simulation and optimisation components in WP6 will be an integrated workflow that utilises both multi-objective optimisation (MOO)

algorithms and multi-attribute decision-making (MADM) strategies. As such, the simulation will be the standard set-up dependent on the algorithm or Grasshopper component used, but weights will be established for the fitness objectives (Key performance indicators - KPIs), in relation to MADM strategies to establish hierarchy and priority.

2. **Results analysis:** The results of the simulations will be analysed through the visualisations available by the selected optimisation algorithm or Grasshopper component but will also integrate an additional process of calculating KPIs through mathematical formulation using the numerical outputs of the Analysis models (Human Comfort Simulation, Environmental Models, and Ecological Model). The results of these calculations will contribute to the optimised fitness values.
3. **Filtering and ranking of outputs:** The filtering process will be conducted through an algorithm that eliminates outputs that do not fit the initial fitness objective thresholds generated in the architectural design phase. The generated range of Pareto solutions (optimised solutions with more than one objective) will then be ranked using a selected MADM strategy using the weights that were established in the design optimisation setup. This algorithm will rank the generated outputs based on the weights of the fitness objectives (KPIs) to showcase the best performing designs in order.

However, the optimisation process includes not only the optimisation of the envelope design, but also the voxel model and the KPIs for each iteration. The optimised values (data and KPIs) will be encoded into the respective voxel cells through the same algorithms employed in the architectural design phase which will then be exported as raster information, if running through an iterative loop, or .csv in the case of a final design selection. Thus, the final outcome of the computational workflow is a selection of envelope design with the corresponding metadata stored in a voxel model.

3.3 Conclusions

The computation framework based on Rhino establishes the core processes supported in the ECOLOPES platform, and provides a roadmap for the integration and deployment of the computational components developed in WP3, WP4, WP5 and WP6. From a system architecture perspective, the framework also identifies and defines the data connections and data exchange mechanisms required for integrating the ECOLOPES platform.

In particular, this exercise identifies two main parts of the ECOLOPES platform with different technical requirements, working asynchronously to a large extent: 1) the data and knowledge management layer, and 2), the design, analysis, and simulation services.

In the coming project period, the design and implementation of both parts can progress in parallel, with connections established through the ontology and the system's data warehouse.

4. THE KNOWLEDGE GENERATION FRAMEWORK AND THE MIMO EXPERIMENT

The acronym MiMo stands for MInimalist MOdel. The MiMo experiment was conceptualised to address the knowledge gap identified by the consortium on our ability to understand and predict how design can drive and can be used by the ontology to drive the development of the *ecolope* ecosystem. The set of ecological processes and causal relationships encapsulated in the ecological model is too complex to enable us to a priori predict the consequences of a change in design on all or part of the *ecolope* ecosystem without testing the design by running the full model. Such an approach is hardly compatible with the idea of the EIM ontology which aims to guide design toward given objectives. The MiMo experiment, therefore, aims to create the knowledge necessary to inform the design process to go toward given ecological and/or human comfort objectives. The MiMo experiment will address this objective by 1) enabling the initiation of architectural and ecological variables to be meaningfully correlated, and 2) serving as an open-ended accumulative knowledge generator for the ECOLOPES project.

In computational terms, the MiMo experiment aims to agilise the consolidation, prototyping, and deployment of the data and knowledge management layer of the ECOLOPES platform, focusing primarily on integrating the elemental ecological model, streamlining their execution, and connecting them with the data warehouse to accumulate their results. The MiMo experiment will run the environmental and ecological models on a continuous and linear variation of input data to populate the project's database and create the necessary data threshold to identify meaningful correlations between architectural or design variables and ecological variables.

4.1 Goals of the MiMo experiment

In its first version, the MiMo experiment will enable us to understand how given building geometries impact the environmental conditions (such as soil depth and water retention, radiation input, and general connectivity of the *ecolope*) and the structure and composition of the ecological communities on the *ecolope.* The objective is therefore to use the experiment to extract general relationships between architecture and ecology. These relationships will then be used as possible architectural options to guide design in the generative design and design optimisation processes of the computational workflow (see Section 3 of the present deliverable).

So far, the knowledge directly linking architecture to ecology exists only in fragmented and specific ways. For instance, we might know from the literature review the needs in terms of food and shelter resources of a given bird species. Such knowledge gives hints to architects to create artificial nests of the right shape and height to potentially attract the target species. However, the artificial nest may actually enable an individual to live on the building only if a number of other conditions are met, i.e., if the environment provides the other factors necessary to the species to complete its life cycle, such as access to enough food, mates, and acceptable probability of death. The MiMo experiment will help to understand how the geometry of the building can help support given functional groups or types of ecosystems while accounting for all these important factors that interact with the species life cycle.

The MiMo experiment relies on the use of the different environmental and ecological models developed or applied in the computational workflow for the development of the platform (see section 3 of the present deliverable). It will simulate the environmental conditions and ecology of *ECOLOPES* designed with a high number of simulated building geometries. These simulated geometries will be built to cover the range of simple possible *ECOLOPES* geometries in a multiscalar approach, including variations going from the microscale (1 to 100 cm scale) to the macro-scale (general shape) of the building.

We envision the MiMo experiment to be extended toward the exploration of other architectural questions, for instance to better understand how to change the geometry of a building to support given functional groups or ecosystems under various climatic conditions, or the role of management.

4.2 General structure of the MiMo experiment

The MiMo experiment is composed of two computational steps. The first step explores how geometry influences the abiotic conditions on the *ecolope*, using a set of environmental models for soil erosion, water retention, radiation, and connectivity (Figure 10). It will contribute to answer simple but important questions regarding the environmental conditions induced by given geometries (and which will then support the *ecolope* ecosystem) such as:

- Can the geometry of the building maintain soil where we place it, or will the soil erode and accumulate in other places?
- Does the geometry of the building enable water to be stored/retained in the soil in some areas, or does it directly flow down?
- How does the geometry of the building modify the input radiations on the *ecolope*?
- To what area (m²) does the geometry of the building allow access for a walking animal?
- Which geometries enable to optimise part or all of these different aspects?

The second step explores how the environmental conditions induced by the geometry can drive the development of the *ecolope* ecosystem and promote certain plant or animal functional groups and no others. The two steps together will enable us to build the necessary knowledge and hopefully generalities on the response of the ecosystem to the building geometry in different environmental contexts.

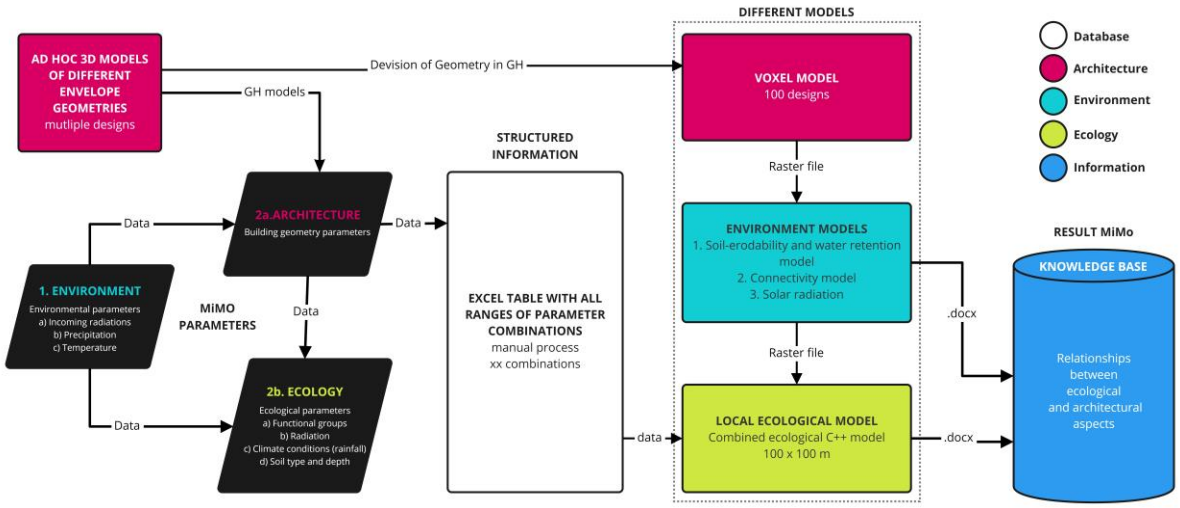

Figure 10: The computational workflow in the MiMo experiment.

4.3 MiMo inputs

The MiMo experiment requires a number of inputs in a given format for the involved models to run. Understanding the influence of some of these input variables identified as important by the consortium, e.g., variables describing the geometry of the building, on the ecological variables is the main objective of the experiment. These important input variables will be used as parameters to be varied and tested in the experiment.

The experiment will allow collecting the environmental and ecological outputs of manifold combinations of parameter values. The combination of tested parameter values will be created to cover the range of possibilities of each chosen parameter, with respect to the values of the other parameters (some combinations might be unrealistic). Some interactions between parameters will also be targeted by the experiment. For instance, we will investigate the influence of the interactions between the different geometry parameters on the ecological variables (Section 4.3.1). This section details the first parameters that will be targeted by the experiment.

From 3D CAD to raster data – geometry parameters: The variables describing the geometry of the building are the first ones targeted by the MiMo experiment. Geometry is expected to influence the environmental conditions on the *ecolope* (e.g., connectivity, radiation input), and therefore the species that can live on it. This section introduces a new method of how we envisioned correlating geometry and the corresponding metadata for a defined analysis grid. Thus, in MiMo, an initial range of geometry typologies (e.g. box, sphere, cylinder, cone, tetrahedron, etc.) were programmed within Rhino's visual programming environment and divided into voxel units of two different sizes: 1 m × 1 m × 1 m, and 10 m × 10 m × 10 m (Figure 11).

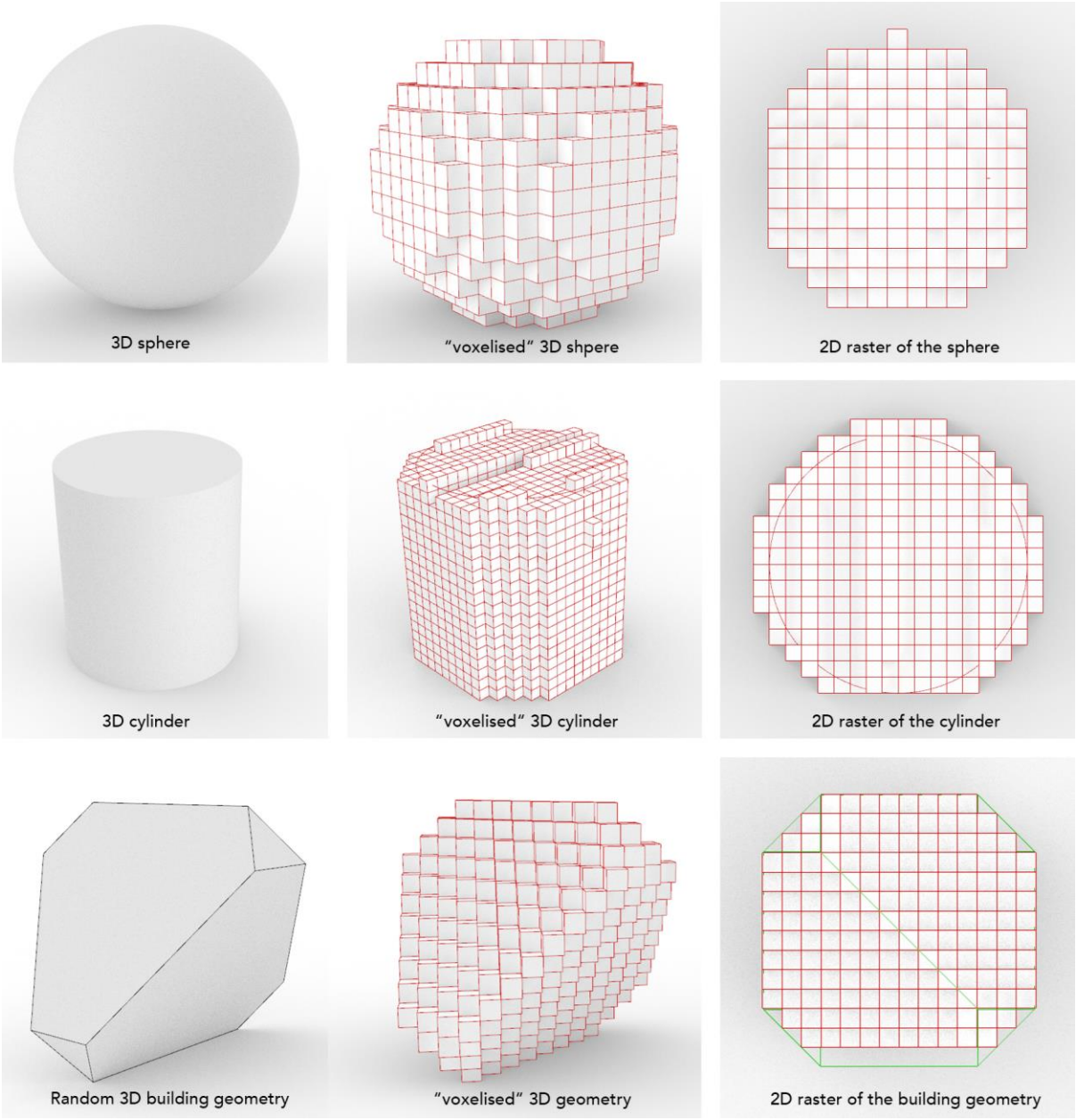

Figure 11: Grasshopper algorithms that generate different building typologies and 'voxelise' them into units.

In the next step, an algorithm extracts relevant geometry parameters' values, i.e., surface roughness, surface area, surface angle/steepness, and building massing. The Rhino environment then enables us to vary the geometry parameter values to cover their range of possibilities, and ultimately explore how their variations impact the ecology (Table 1). They are then stored within a database that can be exported as a .csv table or a set of rasters. The rasters are then used as input to the models (Section 4.4).

Table 1: Geometry parameters to characterise the voxel cell developed by WP5, WP6, and WP7.

Geometry parameters	Description	Range	Units
1. Surface roughness	Deviations of the normal vector of a real surface from its ideal form	0 - 1	Average Roughness (Ra)
2. Surface area	Total area of a single surface		Area (m^2)
3. Surface angle/ steepness	Angle between the surface normal and a reference plane	0 - 180	Degree (°)
4. Building massing	General building shape, form, and size (height and planar dimensions)	High rise - Medium Rise - Low Rise - Single Storey	Volume (m^3)

4.4. MiMo models

In the first computational step of the experiment, four MiMo environmental models compute the soil depth, water retention, solar radiation, and contribution to connectivity of each voxel cell based on given inputs. Building-scale values will be computed to evaluate the general impact of the geometry for each of these environmental conditions (Table 2). In the second step, the local ecological model will use the geometry and the environmental conditions generated in the first step to model the responses of soil development, plants, and animals to the geometry.

Table 2: Environmental parameters computed by the four environmental models to characterise each voxel cell and the overall environmental performances of the geometry (developed by WP3– WP7).

To compute	Description	Range	Units
Soil depth per voxel cell (volume)	Soil remaining/accumulated on a cell after erosion	0-N/A	Volume, mm3, cm3, m3
Connectivity	Modelling of connectivity network based on Graph theory	0-1	Probability of connectivity (per voxel cell and for the entire *ecolope*)
Solar radiation	Solar radiation for a specific geometry (Shadows, heat)	0-5,7 kWh/m2 depending on the geographical location	Global Solar Irradiation (kWh/m^2) Incident radiation
Water retention/Hydrological model	Water fluxes and retention	N/A	Water Retention Value (WRV)

4.3.1 The soil depth model

Soil is required for microbiota, plants, and is used as shelter by some animals. The soil depth model computes how much soil can be placed on and retained by a certain geometry. Spatially-explicit soil erosion modelling, as enabled by the Universal Soil Loss Equation (USLE), usually requires information on run-off, slope, erodibility of the soil, vegetation, and practices (e.g. El Jazouli et al., 2017). A model derived from this equation, also simulating soil accumulation (see Jakubínský et al., 2019 for a model comparison), can easily be applied to the ECOLOPES framework after soil erodibility parameterization, a necessary step because on an *ecolope* the soil will most probably be an artificial substrate and not a natural soil. The model outputs the soil volumes for each voxel cell.

The soil depth model will use a surface model as input. We will test how surface roughness values (Ra) and surface inclination influence the resulting distribution of soil depth over the *ecolope*. In the next step, voxel cells characteristics with the potential to retain soil are selected and the volume of the soil is calculated for each voxel cell (Figure 12 and 13).

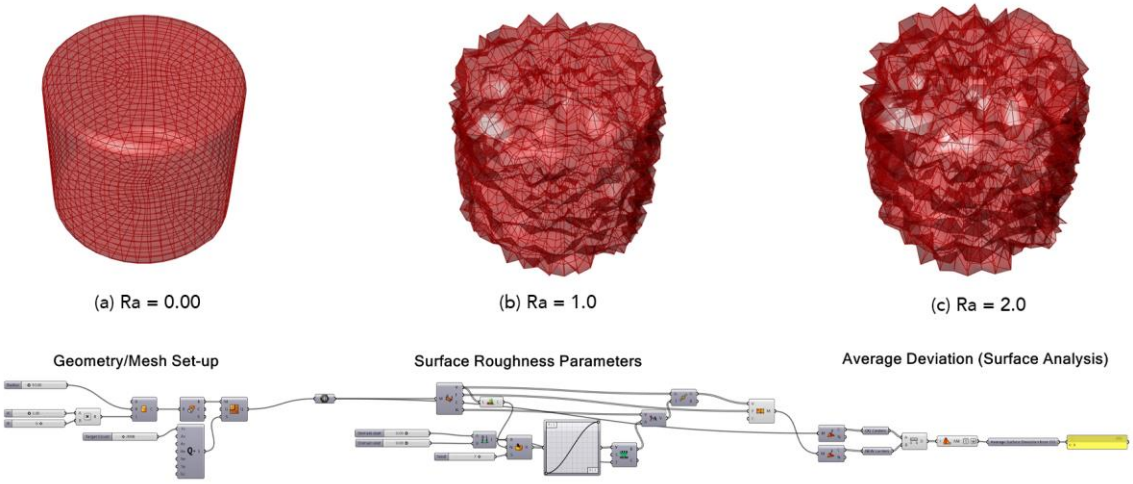

Figure 12: An approach to compute the surface roughness value Ra through a parametric model in Grasshopper.

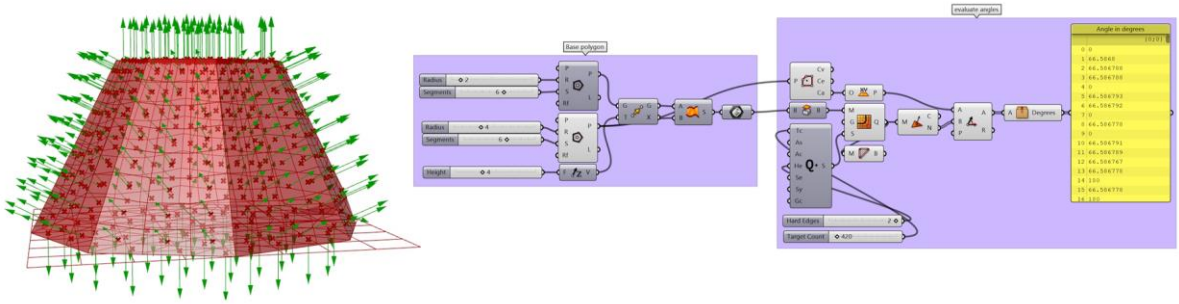

Figure 13: An approach to compute the angle of the surface inclination through a parametric model in Grasshopper.

4.3.2 The connectivity model

The connectivity model proposes a very general approach for connectivity based on graph-theory, where the ecological connectivity network is inferred using the least-cost path approach to connect habitat patches (Graphab: https://sourcesup.renater.fr/www/graphab/fr/home.htm) (Foltête et al., 2012). The connectivity will be computed for a walking animal, defining cells occupied by soil (from the soil depth model) as potential habitat/resource. The resistance of the 3D surface to movements of the animal will depend on the slope between voxel cells. The underlying assumption behind computing a general connectivity network based on a walking animal is that walking animals are the organisms that are more likely to be limited in their ability to move through the 3D surface by steep slopes. Thus, if a walking animal can reach an area, it's very likely that most organisms can reach it.

The connectivity model informs which voxel cells and path(s) can be used by the walking animal for reaching these areas. The outputs are:

- A map giving the probability of connectivity of voxel cells occupied by soil (habitat patches) (PC, Saura et al., 2007)
- A connectivity map based on the number of times a voxel cell is part of a path (corridor function)
- A general probability of connectivity for the entire 3D surface that reflects the probability that two points are taken randomly in the area are connected
- A 3D connectivity grid with the CAD environment

4.3.3 The solar radiation model

Besides soil, light is required for plant growth. Solar radiation model extracts light values based on the input geometry using the Ladybug Tool in Grasshopper. The workflow for the solar radiation simulations is mainly composed of three steps:

Firstly, a location for the simulations has to be set up. Once the location is defined, an .epw file is needed to import climatic data: in this script, the open-source database used is directly connected with Ladybug Tools: https://www.ladybug.tools/epwmap/ (Figure 14). After importing the set of weather data, data concerning solar radiation are available for the simulation.

At this stage, the second step is related to the definition of the envelope shape (coming from the GH script previously defined) and the ground (e.g. planar surface, slope, etc), on which the distribution of solar radiation will be analysed. Finally, the last step is related to the definition of the grid size, depending on the resolution that we want to achieve with simulations (e.g. 1 × 1 m^2 or 5 × 5 m^2). Outputs of this workflow are specific values of incident radiation (kWh/m^2) for each cell of the mesh composed by envelope and ground.

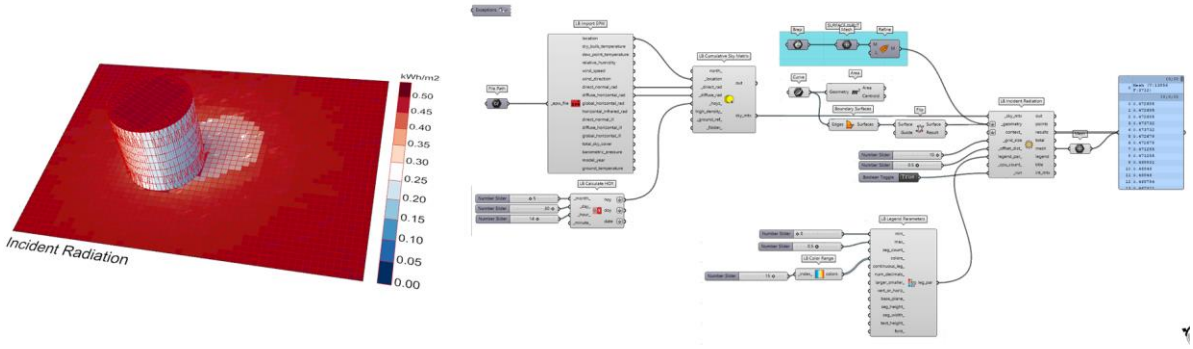

Figure 14: Solar radiation values computed for each voxel cell within the model.

4.3.4 Water retention model

The water retention model computes how much water can be retained by a certain geometrical configuration. As an input, it would need either a Digital elevation model (DEM, a 3D computer graphics representation of elevation data), NURBS, or a polygon mesh model. DEM is often required for flood or drainage modelling. Docofossor, a Grasshopper plugin (Section 2.4.4) uses DEMs as input for analysis (Sun et al, 2020). However, there are existing examples of how water retention models can also be simulated in Grasshopper using NURBS and mesh geometry.

4.3.5 The local ecological model

The ecological model is described in the current deliverable in section 3.2 and more extensively in the deliverable D4.1. In the MiMo experiment, the ecological model will likely be run over a 50 or 100 years' period to simulate in a spatially-explicit way the state of the *ecolope* ecosystem in terms of soil types, plants and animal functional groups distribution. These spatially-explicit outputs will be used to understand how the fine-scale geometry influences the ecological characteristics of the voxel cells. The ecological outputs will also be used to compute building-scale ecological variables meaningful to reflect important ecological processes. For instance, the higher trophic level present on the *ecolope* could be used to reflect if and how the geometry influences the trophic structure of the ecological community.

4.4 The Knowledge base (KB)

The data produced by the MiMo experiment will fill up the knowledge database, where each line contains a unique combination of parameter values and the resulting environmental and ecological outputs. This knowledge base will be used to extract correlations between the tested parameters and the environmental (soil depth, water retention, input radiation, and connectivity) and ecological variables (e.g., distribution of soil types, plants, and animal functional groups, functional group richness). Such a database is crucial as a starting point for

gaining knowledge about the relationships between geometry and urban ecology on an *ecolope*. This knowledge will be then leveraged into the ontology (see D4.1).

4.5 Conclusions

The conceptualisation and design of the MiMo experiment allows consolidating the data and knowledge management layer of the ECOLOPES platform, and elicits its general technical requirements in terms of data, data structures, computation, chaining and integration, and other concerns. By the end of the related technical work, a first stable version of the ecological model will be deployed and integrated with the platform's data warehouse, and mechanisms will be put in place to execute the model autonomously and continuously.

5. SOFTWARE DEVELOPMENT APPROACH FOR ECOLOPES

The software development approach adopted in ECOLOPES takes into account the type of applications, their maturity, and intended usage and lifecycle. Whereas Agile-based methods are best suited for developing mainstream applications and sustaining a healthy software development pipeline, they are not generally well suited for experimental development intended to prototype the results of research and innovation activities. The collaborative and tentative nature of software development in this context also does not fit well with waterfall-based methodologies. Instead, we choose to manage software development based on an incremental approach, in which specific sections or components of the platform are designed, prototyped, integrated, and evaluated in each iteration. This approach also allows us to design and develop several sections in parallel, a process supported by the early definition of the system architecture, common data models and interfacing mechanisms.

Accordingly, an overall system architecture was first designed, facilitating the deployment of the ECOLOPES platform's digital infrastructure and its main components. On top of the infrastructure, a cloud-based environment that supports the deployment of algorithms and geometric computation components that have been developed and deployed (Section 7). This includes data warehousing capabilities and a collaborative environment where different components can be tested and refined. In the current iteration, while this environment is consolidated and leveraged to help develop the project's analytical components, parallel efforts are invested to integrate the ecological models and deploy them, and operate them to generate knowledge as described in section 4.

5.1 Data and process specification for all components

Data and process specification for all components of the platform facilitates a common computational workflow and to work in an agile manner on parallel development efforts. Furthermore, by defining all inputs and outputs, a better understanding between the interdisciplinary development groups can be fostered and development becomes more efficient from the beginning. Table 3 demonstrates an overview of the collected information on components and data specifications to be later integrated into the platform.

Table 3: Overview of the data and process specification for all components.

Component	Description	Inputs	Outputs
ECOLOPES Information Model (EIM ontology) (WP4)	The reasoning framework for the ECOLOPES platform that leverages information from the KB and makes it available for design.	KB	Web Ontology Language (.OWL.)
The abiotic environment and architecture dataset (WP4, WP7)	Georeferenced datasets (abiotic conditions, socio-economic) and local building features (city scale and local scale) as well as information on normative constraints and design that aim to enable a comprehensive description and evaluation of a potential ECOLOPES site and serve as inputs for the modelling and simulation processes.	International and national geo-referenced datasets. EPW, GIS, OSM	Open Database Datasets (WP4 and WP7).
Environmental models (WP3-7)	Georeferenced dataset describing the environment (e.g., incoming radiations and soil depth per cell) as a result of 3D geometry and other environmental variables (e.g., incoming radiations, precipitations).	Open databases, Raster datasets, 3D geometry	Raster datasets
Ecological model (WP4)	Model of the interdependent spatial and temporal dynamics of soil/microbiota, plants and animals, as a response to building geometry, abiotic conditions and substrate. Data to parametrize the model is retrieved from open databases (GBIF, TRY, PREDICTS, BIEN, SoilGrids) and experiments	Raster datasets from open databases and from environmental models	Combined C++ model, Raster datasets
The microbiota dataset (WP4)	The experiments provide data on microbiota composition in different soils that are used by plants and animals. Most importantly, the microbiota model data will provide variables for catalysts for nutrient/carbon cycling and plant growth promotion to establish rates of soil development	Field experiments	Values for soil development, soil model parametrisation.
The human dataset (WP6 and WP7)	Analysed data on human comfort conditions, physiological, psychological and social benefits of nature to humans, with a focus on various health and well-being and comfort outcomes (including ecosystem services).	Literature review, 3D model (3DM/ GH file)	Grasshopper file, metadata stored in a voxel model.
The KB (WP3-7)	Resulting data from the MiMo outcome (Ecological and Environmental Models).	Ecological and environmental model outcome.	Correlations between ecological, environmental and architectural parameters.
The design generation and optimisation environment (WP5 and WP6)	A set of algorithmic tools and processes which will run as a backend service and that will be used by the front-end tools. Based on the design outcome, the toolset will be validated by WP7.	OWL/JSON 3D terrain model, requirements from the stakeholders, fitness objectives	3D model Voxel model .CSV, .JSON

ECOLOPES front-end tools (WP3)	User interface with the developed data-driven recommendation system through two front-end tools based on the Rhino platform. Through the tools, the user can access site-specific real-time data from the data warehouse and algorithms from the design generation and optimisation environment.	GH algorithms were developed by WP5, WP6, WP7. C++ programs developed by WP4.	Grasshopper plugin, A web interface for ECOLOPES design and design-recommendation.
ECOLOPES Multi-Species Habitat (WP7)	Feedback from real-world design cases at the four different sites to validate the developed data-driven Recommendation system from a multi-species perspective, will provide real-world parameters for optimisation (WP6) to achieve more realistic design outcomes.	Real-world prototypes, 3D models in VR	Results from monitoring in VR and from Real-world prototypes.

5.2 Conclusions

The specification of data and processes reveals a broader understanding of how the components can be developed in smaller, or in the case of the Ecological Model (see D4.1) and the Design generation and optimisation environment (WP5, WP6) even in a larger cluster. Data inputs and data outputs are defined and thus, a messaging infrastructure to allow different systems to communicate through a shared set of interfaces can be developed in a parallel process. However, first, the system architecture for building the ECOLOPES platform needs to be designed. It is presented in the following Section 6.

6. THE ECOLOPES SYSTEM ARCHITECTURE

The ECOLOPES system architecture (Task 3.1, ECOLOPES System Architecture) implements the conceptual model of the ECOLOPES platform. It is designed to satisfy the overall functional requirements of individual processes encapsulated in the platform. In order to implement the ECOLOPES platform, cloud-based service-oriented architecture is selected to leverage the scalable resources of cloud computing and to provide the necessary flexibility and modularity for the implemented system to grow and adapt in later stages of development that aim to increase its technological maturity level. The use of cloud architecture to deploy the ECOLOPES platform also enables the configuration of specialised machines for each concern.

Three different components constitute the core of the system architecture: a data warehouse server capable of storing shared data created for/by the system services; a geometric computation environment where specialised geometric computation programs can be deployed as services; and an ecological simulations server for running specialised programs that implement ecological models and non-geometric computation. Together, these three components constitute the backend of the ECOLOPES platform, which support the expert execution of designed workflows in stages.

In addition to these three components, the ECOLOPES platform includes the user machine, which houses the user tools that connect to the backend and use its services. The tools allow general users to incorporate ECOLOPES services in their workflows. The user machine can be

implemented both as a personal computer or a cloud-based server as a shared environment (Figure 15).

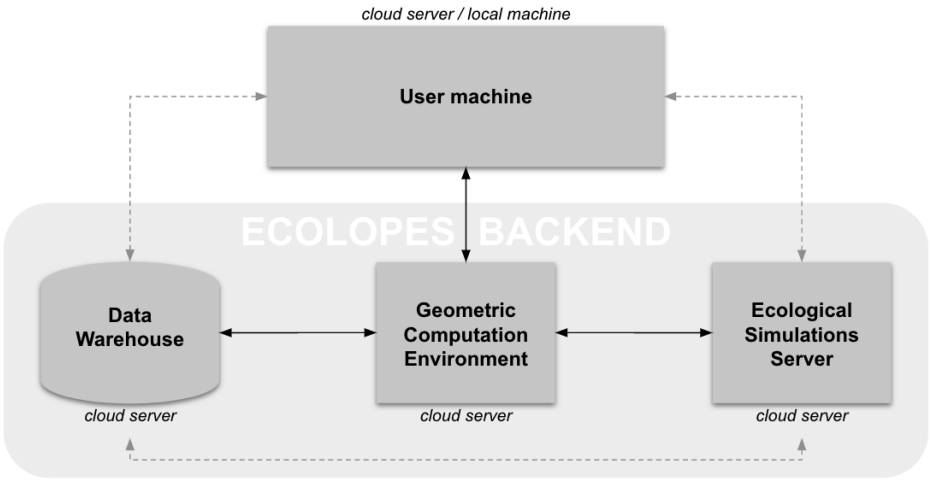

Figure 15: The architecture model of the ECOLOPES platform.

This model of the ECOLOPES system architecture is shown in the following Figure 16. According to this design, the geometric computational environment orchestrates the relation between the user machine and the system backend. It channels requests to the ecological simulations environment and manages the related data input/output from the Data Warehouse. The architecture model also supports direct communication between any of its four conceptual components.

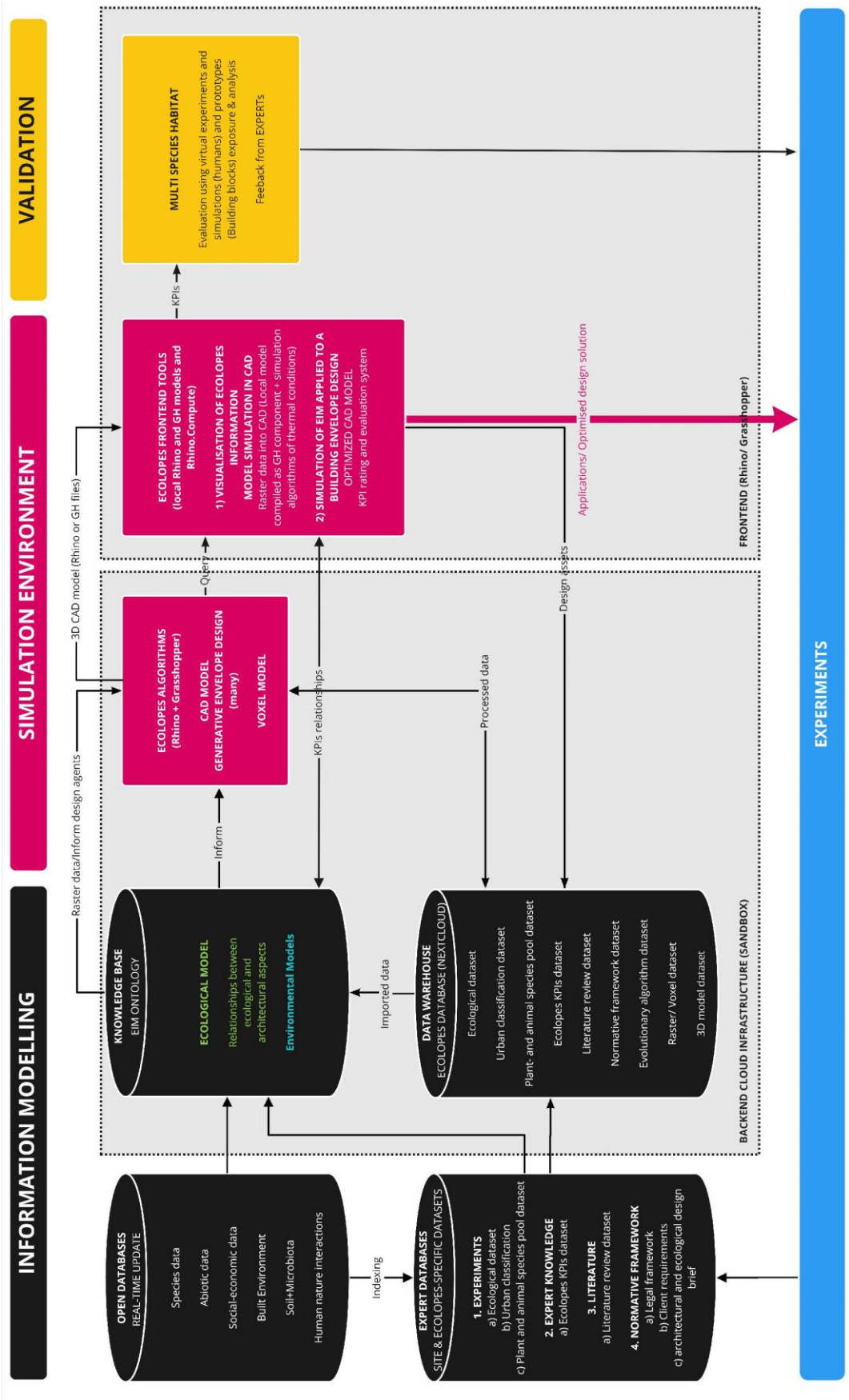

Figure 16: The first version of ECOLOPES functional architecture and its components.

Accordingly, a first version of the functional system architecture was designed to implement the architecture model taking into account the project's development plan. Consequently, the focus of this first version of the system is on data management, deployment of main components (meeting their technical requirements), and establishment of the data exchange pipeline among the system constituents. The design of this first version is shown in the figure above, where components pertinent to the Data Warehouse conceptual module are visualised in black, ecological and computational algorithms and programs are represented in pink and evaluation components in yellow. These components are discussed in the following subsections.

6.1 The software components

The ECOLOPES system architecture has five main components, which are introduced and described in more detail in this section. 1) The EIM ontology with the Ecological Model; 2) the ECOLOPES algorithms, which link the generated datasets of the EIM Ontology with geometry objects; 3) the ECOLOPES data warehouse, which provides data to process the ECOLOPES algorithms; 4) the ECOLOPES computational simulation environment, which iteratively optimises design outcomes based on KPIs; 5) the ECOLOPES front-end tools that enable to interface with the data-driven recommendation system; and ultimately 6) the ECOLOPES Multi-Species Habitat which evaluates the ECOLOPES from the perspective of all inhabitants (plants, animals, microbiota, and humans).

6.1.1 The EIM ontology component

The EIM ontology is a key component for the data-driven recommendation system. It integrates the following five modelling components into one system by modelling its relationships to index and fuse data to form the basis for the development of the ECOLOPES algorithms and the ECOLOPES computational simulation environment. The generated models are interlinked with established feedback loops. All models refer to the same spatial (site-specific) and temporal (monitoring, assessment time) parameters. For a further description of the modelling approach, see D4.1.

6.1.2 The ECOLOPES algorithms component

The goal of the component is to create a link between the EIM ontology and the computational model in Rhino through a voxel model. The computational model is a set of algorithmic and evolutionary computation processes which will run as a backend service and that will be used for the front-end tools (the Rhino plugin). Based on the design outcome, the algorithms will be validated. For a further description of the evolutionary generative design process and the voxel model, see D5.1.

6.1.3 The ECOLOPES data warehouse component

The component stores all ECOLOPES-relevant data and makes it available to the algorithms-, computational simulation environment, and front-end tools components. It also stores geometry data which includes the voxel model, generative design outcomes, 3D analysis and simulation results, and metadata.

6.1.4 The ECOLOPES computational simulation environment component

The component converts the data-integrated computational model (Section 6.1.3) into a computational simulation environment by computational simulations (generative design and optimisation), multi-criteria analysis and rating strategies that enable decision-making processes for the design cases (by defining KPIs and interrelationships/hierarchies between them + expert knowledge); and second, by validating the computational workflow to ensure integration and interoperability through the design cases in preparation of design validation (envelope and building block evaluation).

6.1.5 The ECOLOPES front-end tools component

The component enables users to interface with the developed data-driven recommendation system through two front-end tools based on the Rhino platform. Through the tools, the user can access site specific real-time data from the data warehouse and algorithms from the algorithm component to visualise the simulated output of the EIM Ontology for a specific period of time, and to apply it to a building design at the selected location. The tool recommends a series of evaluated and optimised design outcomes based on the ECOLOPES system to the user that consider the requirements of all inhabitants equally. Thus, it helps the user in the design decision-making process.

6.1.6 The ECOLOPES Multi-Species Habitat component

The component provides feedback from real-world design cases at the four different sites to validate the developed data-driven recommendation system from the perspective of all inhabitants (humans: comfort and well-being; plants + animals + microbes: 12 months Building Block analysis, and by comparing the outcomes for all sites. It will provide parameters to the computation simulation environment component for further optimising to achieve the best design outcome.

6.2 Advantages of the drafted system for the ECOLOPES project

The architecture design applies a separation of concerns between data, geometric computation, and ecological simulations, and addresses the specific technical requirements of each separately. This resulted in the definition of the aforementioned components. Specialised machines can be deployed to support each component, thereby allowing the system to scale, and evolve easily. This also facilitates platform instantiation and deployment in different environments, as well as its seamless integration with other complementary platforms or programs that run on top of Rhino and Grasshopper or that are capable of leveraging the data generated over the analysis and evaluation of specific design cases.

In the short run, the architecture design also allows specialised and parallel development of the platform's data management approach, its ecological models, its geometric computation services, and user tools.

6.3 Technical requirements for building the ECOLOPES platform

To develop a platform architecture for the ECOLOPES data-driven design recommendation system, first the technical requirements from the stakeholders as well as the user of the platform need to be well understood and defined (Ecologists represent the non-human stakeholders). Only a common approach fosters a better understanding between the

interdisciplinary development groups and helps to define the setup and features that the design platform needs to include. Table 4 shows some examples for user requirements from the stakeholders that were translated into technical requirements.

Table 4: Example of user requirements from ecologists and architects for the ECOLOPES design platform.

User requirements (UR)	**Technical requirements (TR)**	**Priority rating** based on MoSCoW framework M-Must/ S-Should/ C-Could/ W-Won't have
UR1: As an **ecologist**, I want to measure the biodiversity (=abundance of FG) for an *ecolope*.	**TR1:** Algorithm and UI to calculate and display count for all species/ FG, output as number, or Shannon Index.	M
UR2: As an **ecologist**, I want to choose/customise the ecological objectives of the *ecolope*, e.g. target specific functional groups (FG).	**TR2:** Enable an animal/plant/microbiota/soil/human/abiotic - aided design. **TR3:** User-driven selection of FG, parameters (Also requires a UI).	M
UR3: As an **ecologist**, I want to choose the best management options to reach the ecological objective for the *ecolope*.	**TR4:** UI to choose management options from, e.g. mowing pattern, to simulate different scenarios.	M
UR4: As an **architect,** I want to choose which information (e.g. soil structure, surface materials, microbiota, plant species, animal habitats, biodiversity, endangered species, climatic conditions, degree of environmental pollution, land-use, building regulations and laws, nature protection) the digital terrain model should contain.	**TR5:** Combine digital elevation model (DEM) of the envelope design (geometry) with the metadata stored in the voxel model. **TR6:** Method for visualising different analysis outcomes, e.g. endangered species (Also requires UI).	M
UR5: As an **architect**, I want to see design iterations in real-time with the KPI values of each design iteration.	**TR7:** Evolutionary computation in real-time will depend on the processing capacity of the cloud-based platform. UI for showing the KPIs and analysis results.	S
UR6: As an **architect**, I would like to select KPIs for each of the stakeholders	**TR8:** KPIs need to be predefined. UI to query from the database the KPIs wanted for an envelope design.	M
UR7: As an **architect**, I would like to visualise the impacts of geometrical manipulation on the KPIs per stakeholder.	**TR9:** Recompute KPI values as feedback for the EIM Ontology and EIM.	S

6.4 Conclusions

By the number of partners and disciplines involved in the ECOLOPES project, diverse datasets, and processes are involved that will have to be first considered, and then integrated into the platform. Thus, the system architecture will have to offer a flexible and scalable solution in order to guarantee a well-adapted service. However, the creation of highly complex modelling approaches such as the ECOLOPES data-driven recommendation system include a proper

exploration and experimentation process which involves the development of preliminary algorithms and combinations of modelling methods for which test datasets and a common shared 'playground' is required. The ECOLOPES sandbox is a preliminary version of the ECOLOPES platform with a more simplified system architecture, which will be described in Section 7.

7. THE SANDBOX – A CLOUD-BASED PLATFORM FOR ECOLOPES

The implementation of the ECOLOPES platform architecture started by deploying its digital infrastructure in the cloud to create an environment where prototypical components can be deployed, evaluated, interconnected, and refined collaboratively. This digital infrastructure is called the **ECOLOPES sandbox**, and provides the necessary computational capacity and resources to execute different types of components, and it is organised according to the system architecture design (Section 6). It implements its three components (data warehouse, geometric computation environment, and ecological simulation environment) prototypically, in a manner that satisfies the minimum technical requirements. At this stage of development, the ECOLOPES sandbox acts as since September 2021 as a fully functioning cloud-based testing environment, a backend, where services and components can be integrated (Task 3.3, Backend development and integration). It can be accessed by interdisciplinary algorithm developers within the consortium, students, and testers.

7.1 The sandbox – the 1st prototype of the computational platform

The sandbox is the 1st prototype of the ECOLOPES computational platform (SO1) in the ECOLOPES project. The system architecture of the ECOLOPES sandbox is rather a simplified version of the ECOLOPES platform, but it outlines how its three main services are connected (Figure 17): 1) Data storage, 2) computation, and 3) algorithm production (in Grasshopper). It is developed and maintained by MCNEEL and fully operating from September 2021.

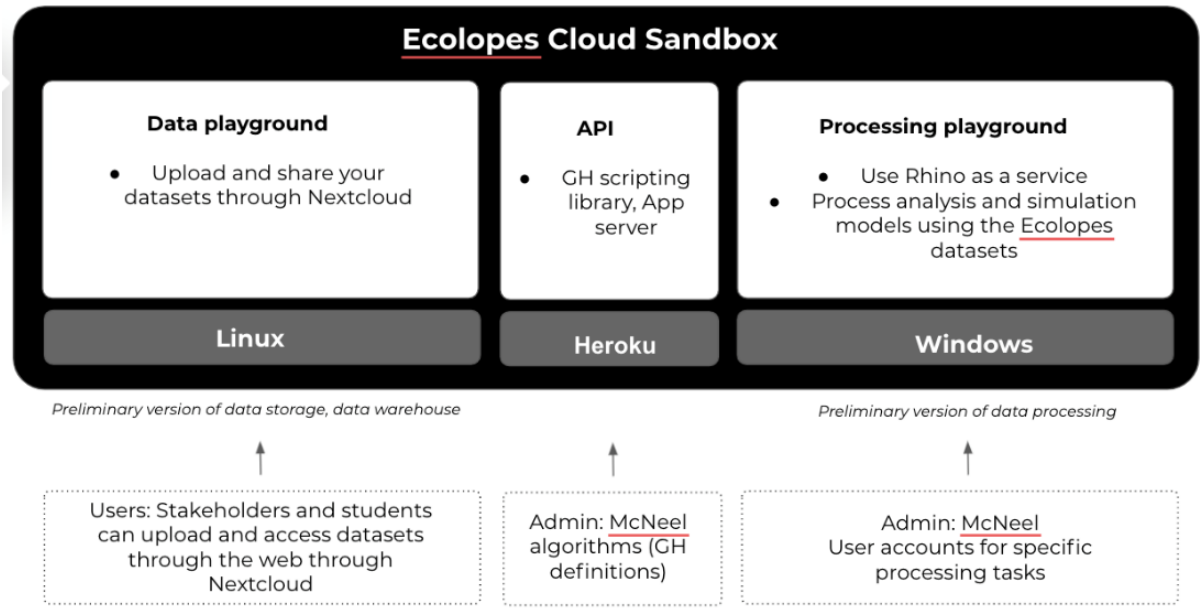

Figure 17: The three main services of the ECOLOPES sandbox.

7.2.1 Data storage

The **data storage server** is a Linux cloud server which can be accessed through NextCloud, a free file sharing and collaboration platform. The server enables file storage and the live exchange of analysed data in a cloud service. The Linux server is a preliminary version of the ECOLOPES data warehousing infrastructure with an existing cloud infrastructure for storing information, especially in relation to WP4, including the ECOLOPES database that includes all data, including spatial-temporal, voxel and 3D models (Task 3.2). The data storage can be accessed through NextCloud: https://data.mcneelresearchprojects.com/ (Figure 18). MCNEEL provided training and documentation for all technical partners on how to use the ECOLOPES data storage for testing and storing datasets (WP4) as well as 3D geometry (WP5).

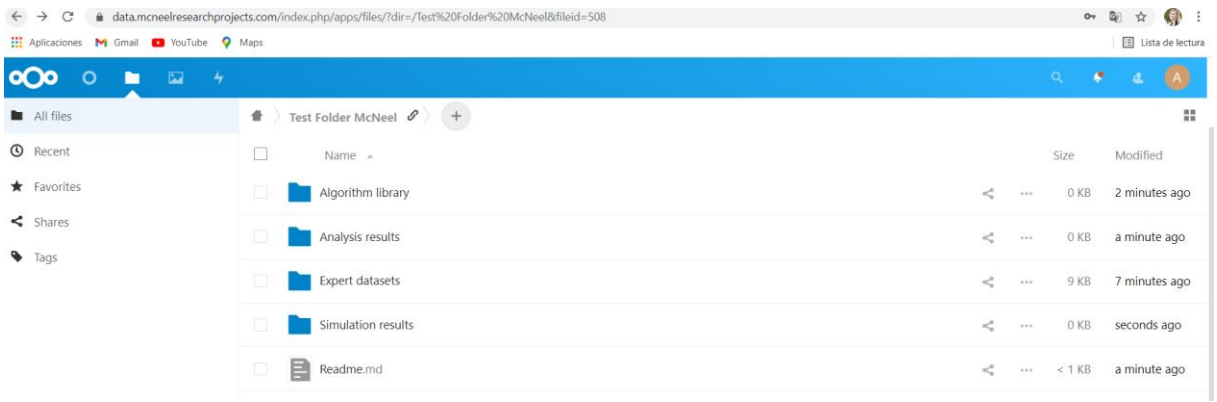

Figure 18: The cloud infrastructure for the ECOLOPES data storage.

Furthermore, the purpose of the NextCloud data storage is:

- Upload and share datasets from individual stakeholders in the cloud
- Upload sample datasets for individual models
- Create an algorithm library for developing the ECOLOPES Algorithms -, computational simulation environment -, and front-end tools components. Upload algorithms for calculation of data processing (GH definitions)
- Save analysis and simulation results, make them available to stakeholders

Furthermore, an SQL database (MariaDB) is hosted in parallel to the NextCloud data storage (Figure 19). This SQL database can be accessed by software components of WP4 implemented in R (D5.1, Figure 17) as well as WP5 software components implemented in Python and integrated with Rhino through the Hops interface (D5.1, section 2.4.3).

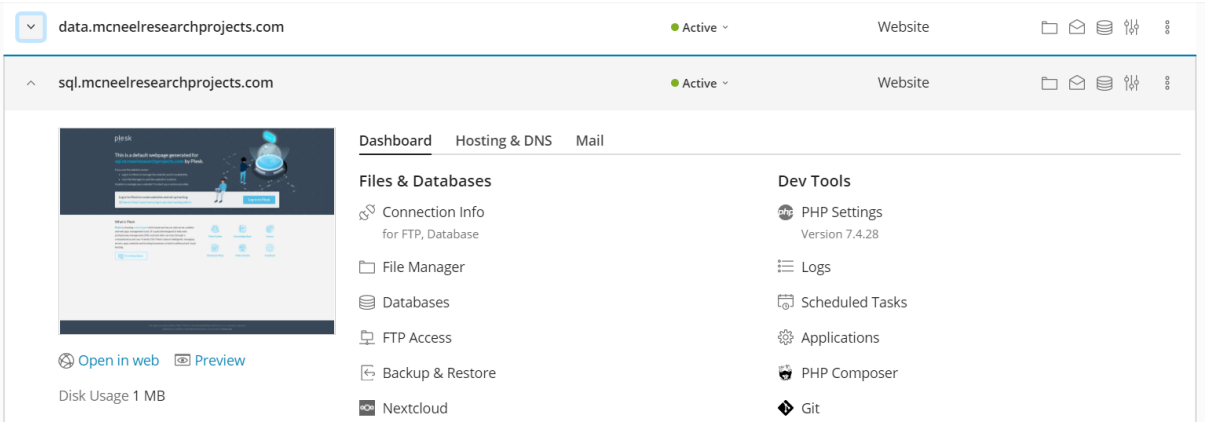

Figure 19: The SQL database is integrated.

7.2.2 Rhino.Compute server

Rhino and Rhino.Compute were chosen as a framework for the development of the ECOLOPES platform (Section 2.4.2). In September 2021, a **Rhino.Compute server** for ECOLOPES was deployed by MCNEEL (Figure 20). Technically, it is a Windows cloud server where Windows Server 2019 runs on a virtual machine. It provides a user interface for communication with local machines through the Grasshopper Hops components (Section 2.4.3), and through the Rhino.Compute AppServer (Section 2.4.2) that displays the computed geometry and data in a standard web browser. Another advantage of cloud computing with Rhino.Compute is that computationally heavy analysis and simulation models (WP5 and WP6) can be processed. Guidelines on how to connect to the ECOLOPES Rhino.Compute server for processing Grasshopper algorithms were documented and communicated to all technical partners. The server can be accessed through the following link: compute.mcneelresearchprojects.com.

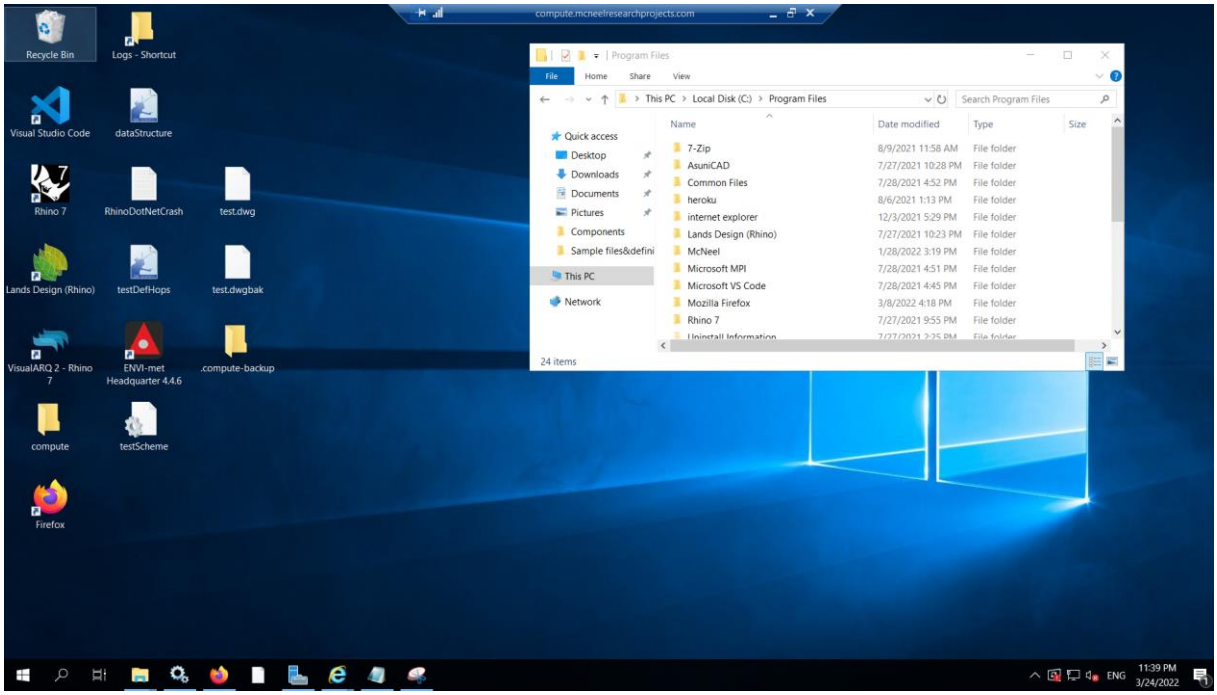

Figure 20: The ECOLOPES Rhino.Compute server, a framework for the development of the design generation and optimisation environment (Section 3.5).

7.2.3 The algorithm production server

The Heroku server stores Grasshopper algorithms to be computed by Rhino.Compute and the Rhino.Compute.AppServer servers. Thus, WP3–WP7 can upload their custom algorithms to Heroku. This Algorithm production server is a dynamic library for all scripts related to 3D geometry objects based on the Rhino platform (Figure 21).

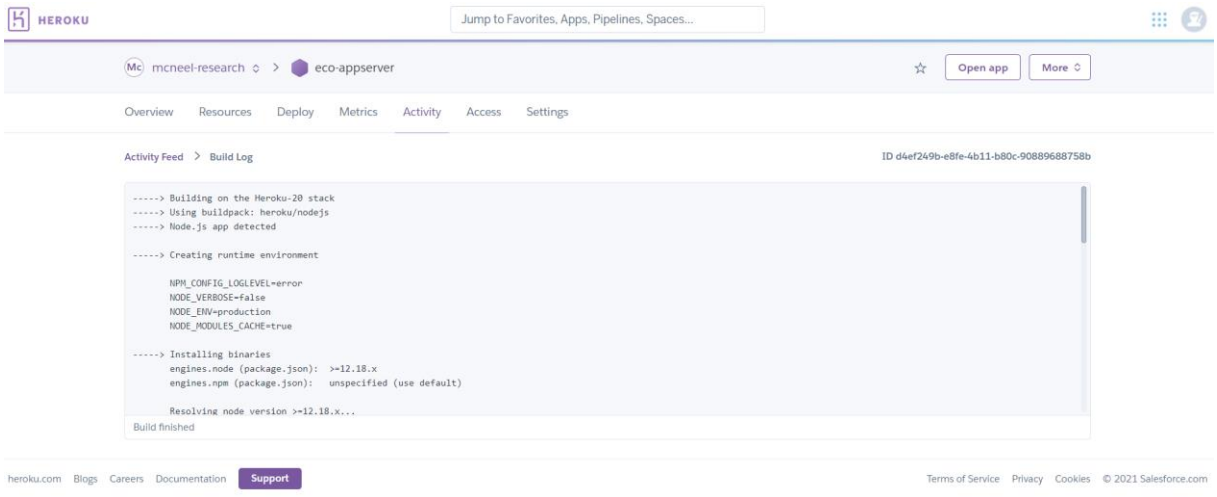

Figure 21: The Heroku server makes GH algorithms or Hops components available on your desktop/ they are computed remotely.

7.2 Technical details on the sandbox setup

The Sandbox is hosted on a company-owned cloud server infrastructure by a private web hosting company in Frankfurt am Main (https://webhoster.de/webhosting/). In contrast to

AWS, all servers are located in Germany and technical support is provided by a small-size trained team. The sandbox can be scaled and adapted to the storage and processing requirements in the ECOLOPES project. Table 5 provides an overview of the setup. Besides the technical details, it includes the financial implications for the development of a new design platform.

Table 5: The Sandbox setup.

Component	Description	Set-up	Costs in EUR (since September 2021)
1. Data storage server	Linux server with NextCloud for all partners.	Storage: 400 GB	28,75/ month
2. Rhino.Compute server	Windows Cloud Server. Windows Server 2019 license.	CPU: 4 cores (64-bit) Storage: 400 GB RAM: 16 GB	56,35/ month 1195,00 (once)
3. The algorithm production server	Heroku server.	-	US 14,00/ month
Domain	The domain is automatically connected to the server.	-	15,00/ year

7.3 Testing of the sandbox: ECOLOPES plugin for Grasshopper

The ECOLOPES front-end tool is a **free Grasshopper plugin** developed by SAAD, UNIGE, VIE, TEC, and led by MCNEEL (Task 3.4, Frontend development). Its aim is to generate, analyse and optimise geometry models of envelope designs to gain knowledge about the ecological performance of the design outcome and its impact on the city. Thus, the plugin contains Grasshopper components for form generation from algorithms that will be developed by WP5 and WP6, and for environmental and ecological analysis developed by WP3-WP7. Further, the plugin will include new Grasshopper components that bridge the gap between raster and geometry data, such as the 'Voxeliser' and the 'Rasteriser´ (developed by WP3) (Figure 22). Lastly, the Expert database and the KB, hosted on the Linux cloud storage setup, can be accessed by specific Grasshopper components developed by MCNEEL. Also, geometry and analysis data can be stored and sent back to the KB.

There are four groups of Grasshopper components for the plugin: 1) Components for data exchange, 2) components for form generation, 3) components for analysis including ecological analysis, solar radiation, soil depth, water retention and connectivity as well as components that can voxelize geometry models and export it as raster data, 4) components for KPI simulation (filtering, ranking, correlations) and optimisation, and 5) preview components that visualise data for each inhabitant and display the final *ecolope* design.

Grasshopper components are written in the C# or Iron.Python language. These components and other algorithms can be compiled as Hops components which allow interfacing with the

implemented Rhino.Compute platform. Using Hops allows outsourcing resource-intensive processes which are driven from your local machine.

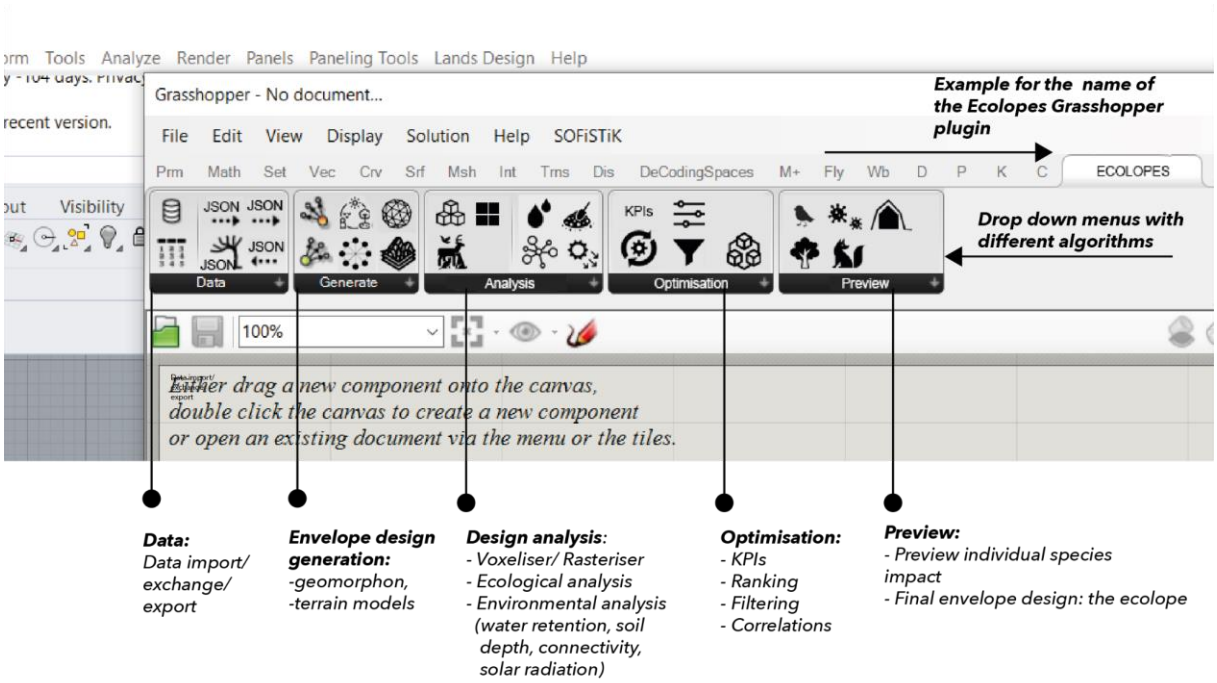

Figure 22: Example how the ECOLOPES plugin for Grasshopper could look like.

8. CONCLUSIONS AND RECOMMENDATIONS FOR THE NEXT VERSION

This report has described the technical requirements associated with the ECOLOPES platform development and deployment. It has showcased the state-of-the-art of similar platforms with related applications and discussed their capabilities with respect to comparable functionalities.

It has also described the design of the computational workflow that specifies how different components of the platform are chained, and how data needs to be managed to support the processes inherent in the ECOLOPES approach.

It discussed how the ecological models are integrated to create the platform's knowledge generation framework, which enables the collection and exploitation of fundamental data pertaining to the relationship between architecture and ecology.

Based on the design of the computational framework and the knowledge generation framework, the requirements in terms of data exchange and interfacing mechanisms were elicited and documented. In addition, the overall architecture model was defined to support a scalable and flexible system capable of supporting the research activities conducted by experts as well as the resolution of use cases brought forward by architects and designers.

The architecture was prototypically implemented under the Sandbox, which deployed all the necessary digital infrastructure required, and provided an environment that facilitates the deployment, testing, and evaluation of computational and analytical components.

In summary, the major achievements in terms of eliciting the requirements of the ECOLOPES platform and prototyping it can be described in the following: Eliciting the requirements of the digital infrastructure and system architecture, and deploying a fully functional production-level infrastructure on top of which the platform can be easily installed. Designing the data pipeline across the different components of the platform, including input/output interfaces and data exchange mechanisms. Consequently, a data management layer that supports data storage, service, and exchange was implemented. In addition, progress was achieved on ecological model integration, through the MiMo experiment.

In the coming phase, the development and deployment of the ECOLOPES platform will build on top of the technical requirements, technical designs, digital infrastructure, and functional prototypes developed so far in the project, in order to deploy a first complete version of the platform, which connects its components more systematically end-to-end.

In the short run, the integration of the ecological models and their functional deployment and connection with the data management components of the architecture will be realised. This will be accompanied by the creation of a first elaborate version of the ECOLOPES data model that governs how data is exchanged between ecological and geometrical components. In parallel, we will aim for a first stable deployment of analytical algorithms and programs that support the design, analysis and evaluation of envelopes, on top of the Sandbox environment. This will support the execution of the ECOLOPES design, analysis, and optimisation processes end-to-end.

References

Research Paper:

Duering, S., Chronis, A., & Koenig, R. (2020). Optimizing urban systems: integrated optimization of spatial configurations. In *Proceedings of the 11th Annual Symposium on Simulation for Architecture and Urban Design* (*SimAUD '20*). Society for Computer Simulation International, San Diego, CA, USA, Article 74, 1–7. https://dl.acm.org/doi/10.5555/3465085.3465159.

El Jazouli, A., Barakat, A., Ghafiri, A., El Moutaki, S., Ettaqy, A., & Khellouk, R. (2017). Soil erosion modeled with USLE, GIS, and remote sensing: a case study of Ikkour watershed in Middle Atlas (Morocco). *Geoscience Letters, 4*(1). https://doi.org/10.1186/s40562-017-0091-6

Foltête, J. C., Clauzel, C., Vuidel, G., & Tournant, P. (2012). Integrating graph-based connectivity metrics into species distribution models. *Landscape Ecology, 27*(4), 557–569. https://doi.org/10.1007/s10980-012-9709-4.

Hurkxkens, I., & Bernhard, M. (2019). Computational terrain modeling with distance functions for large scale landscape design. *Journal of Digital Landscape Architecture, 2019*(4), 222–230. https://doi.org/10.14627/537663024.

Hurkxkens, I., & Munkel, G. (2014). Speculative Precision: Combining Haptic Terrain Modelling with Real-Time Digital Analysis for Landscape Design. *Peer Reviewed Proceedings of Digital Landscape Architecture 2014 at ETH Zurich,* (November), 399–405.

Jakubínský, J., Herber, V., & Cudlín, P. (2019). A comparison of four approaches to river landscape delineation: The case of small watercourses in the Czech Republic. *Moravian Geographical Reports, 27*(4), 229–240. https://doi.org/10.2478/mgr-2019-0018.

Moore, I. D., Grayson, R. B., & Ladson, A. R. (1991). Digital terrain modelling: A review of hydrological, geomorphological, and biological applications. *Hydrological Processes, 5*(1), 3–30. https://doi.org/10.1002/hyp.3360050103.

Seah, I., Masoud, F., Dias, F., Barve, A., Ojha, M., & Mazereeuw, M. (2021). Flux.Land: A data-driven toolkit for urban flood adaptation. *Journal of Digital Landscape Architecture*, 2021(6), 381–392. https://doi.org/10.14627/537705034.

Sun, H., Lee, J., Chen, X., & Zhuang, J. (2020). Estimating soil water retention for wide ranges of pressure head and bulk density based on a fractional bulk density concept. *Scientific Reports, 10*(1). https://doi.org/10.1038/s41598-020-73890-8.

Technologies:

CityPLAIN (2021). A cloud computing tool for urban planning. [Online]. Available: https://www.cityplain.com/. [Accessed: 2-March-2022].

Flux.Broward.Land (2021). Planning for uncertainty preparedness, mitigation & resilience. [Online]. Available: https://broward.flux.land/. [Accessed: 23-March-2022].

InFraReD (2021). Intelligent Framework for Resilient Design. [Online]. Available: http://infrared.city/. [Accessed: 23-March-2022].

InFraReD Tutorial (2021) [Online]. Available: https://www.youtube.com/watch?v=DTiITFXPHOk&feature=emb_logo. [Accessed: 4-March-2022].

KPF UI (2022). KPF urban interface for urban data analytics for informed decision making in the design of buildings and cities. [Online]. Available: https://ui.kpf.com/. [Accessed: 21-March-2022].

Scout.Build (2021). Spatial and temporal urban data analytics for informed decision making in the design of buildings and cities by KPF. [Online]. Available: https://scout.build. [Accessed: 23-March-2022].

Scout.build example (2021). [Online]. Available: https://scout.kpfui.dev/?project=hangzhouhttps://scout.kpfui.dev/?project=hangzhou. [Accessed: 23-March-2022].

Swarm (2021). [Online]. Available: https://swarm.thorntontomasetti.com/. [Accessed: 23-March-2022].

Software libraries and tools:

DeCodingSpaces Toolbox (2017). Computational analysis and generation of street network, plots and buildings. [Online]. Available: https://toolbox.decodingspaces.net/. [Accessed: 20-March-2022].

Docofossor (2019). A terrain modeling plugin for Rhino and Grasshopper. [Online]. Available: https://www.food4rhino.com/en/app/docofossor). [Accessed: 18-March-2022].

Grasshopper by David Rutten (2008). [Online]. Available: https://www.grasshopper3d.com/. [Accessed: 23-March-2022].

Grasshopper component (2018). [Online]. Available: https://developer.rhino3d.com/guides/grasshopper/what-is-a-grasshopper-component/. [Accessed: 17-February-2022].

GitHub Rhino.Compute McNeel (2021). REST geometry server based on RhinoCommon and Grasshopper [Online]. Available: https://github.com/mcneel/compute.rhino3d. [Accessed: 10-March-2022].

GitHub Rhino.Inside (2021). [Online]. Available: https://github.com/mcneel/rhino.inside. [Accessed: 14-February-2022].

Hops component McNeel (2021). [Online]. Available: https://developer.rhino3d.com/guides/compute/hops-component/. [Accessed: 5-March-2022].

Iron Python (2009). [Online]. Available: https://ironpython.net/. [Accessed: 21-March-2022].

Ladybug Tools (2013). Free environmental design knowledge and tools. [Online]. Available: https://www.ladybug.tools/. [Accessed: 23-March-2022].

OpenFOAM (2004). A free and open source CFD software developed primarily by OpenCFD Ltd. [Online]. Available: https://www.openfoam.com/. [Accessed: 2-March-2022].

Rhino 7 McNeel (2021). [Online]. Available: https://www.rhino3d.com/7/new/. [Accessed: 12-February-2022].

Rhino.Compute Guides McNeel (2021). [Online]. Available: https://developer.rhino3d.com/guides/#compute. [Accessed: 1-March-2022].

Rhino.Compute AppServer (2021). A node.js server acting as a bridge between client apps and private compute.rhino3d servers. [Online]. Available: https://github.com/mcneel/compute.rhino3d.appserver. [Accessed: 19-March-2022].

RhinoCommon, 2018 https://developer.rhino3d.com/guides/rhinocommon/what-is-rhinocommon/. [Accessed: 22-March-2022].

Rhino Development (2018). Rhino and Grasshopper Developer Documentation. [Online]. Available: https://developer.rhino3d.com/. [Accessed: 5-February-2022].

Rhino.Inside McNeel (2021). Rhino and Grasshopper inside 64 bit applications for Windows. [Online]. Available: https://www.rhino3d.com/features/rhino-inside/. [Accessed: 14-March-2022].

Rhino.Inside.Revit McNeel (2021). Rhino and Grasshopper inside Revit. [Online]. Available: https://www.rhino3d.com/inside/revit/beta/. [Accessed: 11-March-2022].

Rhino SDK (2018). The Rhino C/C++ Software Development Kit. [Online]. Available: https://developer.rhino3d.com/guides/cpp/what-is-the-cpp-sdk/. [Accessed: 14-January-2022].

Rhino.Python (2018). [Online]. Available: https://developer.rhino3d.com/guides/rhinopython/what-is-rhinopython/. [Accessed: 5-March-2022].

Wallacei (2020). An Evolutionary Multi-Objective Optimisation and Analytic Engine for Grasshopper. [Online]. Available: https://www.wallacei.com/. [Accessed: 23-March-2022].

Appendix

A: ECOLOPES - Component_ Data Specification - All partners.xlsx