D6.6 Final Report on Requirements, Components and Workflow Integration

Document Identification			
Status	Final	**Due Date**	30/11/2021
Version	1.0	**Submission Date**	09/12/2021

Related WP	WP6	**Document Reference**	D6.6
Related Deliverable(s)	D3.4, D3.5, D4.3, D4.4, D5.6, D5.7, D6.1, D6.2, D6.3, D6.4, D6.5	**Dissemination Level (*)**	PU
Lead Participant	KNOW	**Lead Author**	Mark Kröll
Contributors	USTUTT, BUL, PLUS, SZE, PSNC, ATOS, ICCS, ECMWF, MOON	**Reviewers**	Dennis Hoppe (USTUTT)
			Zoltán Horváth (SZE)

Keywords:

Requirements Analysis, Artificial Intelligence, Workflows, Component Integration, Data Integration

Document Information

List of Contributors	
Name	Partner
Mark Kröll, Bernhard Geiger, Christoph Schweimer, Manuela Rauch	KNOW
Sergiy Gogolenko, Dineshkumar Rajagopal, Leyla Kern	USTUTT
Alexis Mandalios, Konstantinos Nikas	ICCS
Krzesimir Samborski, Lukasz Szustak, Piotr Dzierzak	PSNC
Francisco Javier Nieto	ATOS
Derek Groen, Yani Xue	BUL
Robert Elsässer, Christine Gfrerer	PLUS
Zoltán Horváth	SZE
Milana Vuckovic	ECMWF
Amor Messaoud	MOON

Document History			
Ver.	**Date**	**Change editors**	**Changes**
0.1	11/06/2021	M.Kröll	Created initial document, initial table of contents.
0.2	20/09/2021	M.Kröll, S.Gogolenko	Table of content and assignment updates; prepared chapter 4 for partner contributions.
0.3	01/10/2021	B.Geiger, C.Schweimer, M.Kröll	Prepared chapter 3 for partner contributions.
0.4	27/10/2021	S.Gogolenko	Merged partner contributions for chapter 4.
0.5	05/11/2021	B.Geiger, M.Kröll, S.Gogolenko	First complete draft.
0.6	18/11/2021	C.Schweimer, B.Geiger, M.Kröll, S.Gogolenko	Sent for internal review.
0.8	01/12/2021	C.Schweimer, B.Geiger,	Integrating feedback from internal review.

Document History			
Ver.	**Date**	**Change editors**	**Changes**
		M.Kröll, S.Gogolenko	
0.9	02/12/2021	M.Kröll	Version for QM endorsement.
1.0	04/12/2021	M.Lawenda	Final Version

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Mark Kröll (KNOW)	02/12/2021
Quality manager	Marcin Lawenda (PSNC)	04/12/2021
Project Coordinator	Francisco Javier Nieto (ATOS)	09/12/2021

Table of Contents

List of Tables

List of Figures

Alphabetical List of Acronyms

Abbreviation / acronym	Description
3D	Three-dimensional
ABM	Agent-Based Model
ABMS	Agent-Based Model Simulation
ACLED	Armed Conflict Location and Event Data Project
AD/SCM	Automated Deployment and Software Configuration Management
AI	Artificial Intelligence
AIS	Artificial Intelligence Support
API	Application Programming Interface
ARH	Adaptive Recognition Hungary
Auto-ARIMA	Auto Regressive Integrated Moving Average
BO	Business Objectives
BVB	Borussia Dortmund
CAMS	Computer Aided Manufacturing
CAVE	Cave Automatic Virtual Environment
CDR	Call Detail Records
CDS	Climate Data Storage
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
CI/CD	Continuous Integration and Continuous Delivery
CKAN	Comprehensive Knowledge Archive Network
CO	Community Objectives
CoE	Centre of Excellence
COV	Epidemiology
COVID-19	Coronavirus Disease 2019
COVISE	COllaborative VIsualization and Simulation Environment
CSV	Comma-separated Values
Dx.*y*	Deliverable *x.y*

Abbreviation / acronym	Description
ECML	European Conference on Machine Learning
ECMWF	European Centre for Medium-Range Weather Forecasts
EWCloud	European Weather Cloud
FACS	Flu and Coronavirus Simulator
FPOE	Freiheitliche Partei Österreichs
GridFTP	Grid File Transfer Protocol
GUI	Graphical User Interface
HLRS	The High Performance Computing Center Stuttgart
HPC	High Performance Computing
HPDA	High Performance Data Analytics
HPII	HPC/HPDA Infrastructure and Implementation
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification
IPC	Integrated Food Security Phase Classification
ITSM	Information Technology Service Management
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KPM	Kernel Polynomial Method
MAE	Mean Absolute Error
MIG	Migration
MK	Hungarian Public Road
ML	Machine Learning
MOON	MOONSTAR
MOP	Multi-Objective Optimization Problem
MPI	Message Passing Interface
Mx	Month *x*
NEOS	Das Neue Österreich und Liberales Forum
NN	Neural Network
OAuth2	Open Authorization

Abbreviation / acronym	Description
OS	Operating System
OSM	Open Street Map
OSRM	Open Source Routing Machine
PDF	Portable Document Format
PIL	Pilots
POD	Proper Orthogonal Decomposition
POR	Portal
PSNC	Poznań Supercomputing and Networking Center
RDFGraph	Resource Description Framework Graph
REST	Representational State Transfer
REQ	Requirement
SCO	Scenario
SCP	Secure copy protocol
SEIRD	Susceptible, Exposed, Infected, Recovered, Dead
SIR	Susceptible-Infected-Recovered
SLA	Service Level Agreement
SN	Social Networks
SO	Scientific Objective
SOA	Service Oriented Architecture
SSO	Single Sign-On
SVR	Support-vector regression
SZE	Széchenyi István University
TO	Technology Objectives
TOSCA	Topology and Orchestration Specification for Cloud Applications
UAP	Urban Air Pollution
UN	United Nations
UNHCR	United Nations High Commissioner for Refugees
URL	Uniform Resource Locator
VECMA	Verified Exascale Computing for Multiscale Applications

Abbreviation / acronym	Description
VM	Virtual Machine
WCDA	Weather and Climate Data API
WP	Work Package
YAML	YAML Ain't Markup Language

Executive Summary

D6.6 represents the final report of WP6 concluding its developments. This report presents (i) the final set of requirements and their KPIs, (ii) the Artificial Intelligence (AI) enabled use case workflows, as well as highlights (iii) the final outcomes of the integration process. It should be noted that all respective objectives have been successfully completed.

WP6 is in charge of capturing, maintaining and evaluating the requirements over the project's lifetime. These requirements form a common ground for developments and a shared baseline across all work packages, technical as well as non-technical. By establishing a Requirements Management Process, the consortium ensured agility to account for dynamic, unforeseen situations exemplified by the Covid-19 pandemic. D6.6 reports on the status of these requirements in M36 of the project.

WP6 is responsible for one of the cornerstones of the project, AI-enabled use case workflows. The evaluation of state-of-the-art AI methodologies, as well as the actual integration of AI solutions was guided and supported by the domain expertise of HiDALGO's use case partners. Year 3 focused on providing support for HPC simulations (i) by a data-driven estimation of model and simulation parameters; (ii) by reducing the complexity of the simulation model to improve scalability as well as to reduce the overall training time. In addition, D6.6 provides an overview of explored, applied as well as adapted state-of-the-art AI methodologies to support HiDALGO's use case workflows resulting in seven AI components, i.e.,

- Location Graph Extraction and Modelling Environmental Factors for the Migration UC,
- Traffic Forecasting and Proper Orthogonal Decomposition for the Urban Air Pollution UC,
- Community Detection, Synthetic Graph Generation and Synthetic Message Generation for the Social Networks UC,

Last but not least, WP6 takes care of the project's component and data integration by setting up an integration infrastructure (in conjunction with WP5) to guarantee stable and well-integrated software outcomes. In particular in year 3, a unified approach for integrating software of all use cases was designed and implemented. Ansible and Spack are used for automated deployment (and installation) of tools on all HPC clusters and testbeds of the project's infrastructure. In terms of data integration, new static datasets such as sensory data were integrated with CKAN. Finally, mechanisms were installed to automatically deploy and test portal components.

1 Introduction

This deliverable concludes WP6, the "Requirements Evolution and Component Integration" and presents the final set of requirements and their KPIs, the Artificial Intelligence (AI) enabled use case workflows as well as the outcomes of the components' integration. Objectives of this work package were tripartite:

1. The requirements for HiDALGO were captured and evaluated to form a common ground for developments and a shared baseline between all work packages. A requirements management process has been established in the first year of the project to account for dynamical changes in the project.
2. This work package has taken care of the implementation of one of the cornerstones of the project, namely AI-enabled use case workflows. Over the course of the project, state-of-the-art AI methodologies have been explored to support the application execution workflows by working in close cooperation with the use case partners.
3. WP6 also has taken care of the project's component and data integration by setting up an integration infrastructure (in conjunction with WP5) to guarantee stable and well-integrated software outcomes.

All these objectives have been successfully achieved as it is documented in this deliverable. Please note that the project has been extended by three months (M39) and this deliverable is to be submitted in M36 of the project. Still ongoing activities including not yet finished requirements are emphasized in the respective chapters; respective results will be presented at the final review.

1.1 Purpose of the document

The purpose of this deliverable is twofold: first, it provides an overview of the outcomes of year 3 of the HiDALGO project and thereby concludes WP6. In particular, (i) it reviews the requirements evolution process by providing details on removed, added and ongoing requirements. As done in D6.5 [16], it continues to report progress (ii) on the development of AI methods supporting the use cases and (iii) on integrating the developments of technical work packages (WP3, WP4, WP5 and WP6 itself) into an entire solution - into a Centre of Excellence accessible via our HiDALGO portal.

Second, it concludes the journey of supporting the use case workflows by experimenting with, applying and adapting state-of-the-art approaches as well as methods from the field of AI – that being one of the key goals of the HiDALGO endeavour. Three years ago, D6.3 [14] can be considered as the starting point where various ideas and ways of AI support were outlined and were started to be explored. Over the project's lifetime, concrete algorithms were developed, implemented and evaluated in close collaboration with the use case partners; some of them were discontinued as well in case they were not able to contribute to the use case. All these efforts and developments particularly contribute to achieving the project's Scientific Objective Nr. 03 "Introduction of Artificial Intelligence assisted workflows to improve application lifecycle handling" and its corresponding key performance indicator groups.

1.2 Relation to other project work

In our reporting efforts, we seek to avoid duplication of the content in our deliverables, and therefore systematically refer to other deliverables, which are closely related:

- D3.4 (Intermediate Report on Benchmarking, Implementation, Optimisation Strategies and Coupling Technologies) [4], D4.3 (Implementation Report of the Pilot Applications Year 2) [7] and D6.5 (Intermediate Report on Requirements, Components and Workflow Integration) [16] for further information about data integration via CKAN.
- D5.7 (Final Portal Release and System Operation Report) [10], D4.4 (Final Implementation Report of the Pilot and Future Applications) [8], D6.2 (Workflow and Services Definition) [13] and D6.5 (Intermediate Report on Requirements, Components and Workflow Integration) [16] for details on the use case workflows as well as the system workflow.
- D5.6 (Second HiDALGO Portal Release and System Operation Report) [9] and D5.7 (Final Portal Release and System Operation Report) [10] for more details on portal components.
- D6.1 (Requirements Process and Results Definition) [12] for the capturing process and the initial set of requirements and D6.4 (Initial Report on Requirements, Components and Workflow Integration) [15] and D6.5 (Intermediate Report on Requirements, Components and Workflow Integration) [16] for respective updates.
- D6.3 (Artificial Intelligence Approach) [14] for approaches to support use cases with concepts as well as methods from the area of Artificial Intelligence (AI).

This document is the third and last of a series of reports focusing on requirements, algorithmic development of AI components as well as components and data integration (D6.4 [15], D6.5 [16], and D6.6). It reviews the state of the requirements in the 3rd year of the project initially introduced in D6.1 [12], which were then updated in both D6.4 [15] and D6.5 [16].

While D6.5 [16] still presented changes in the workflows of the use cases, final use case workflows are now reported and described in more detail in D4.4 [8]. D6.6 puts an emphasis on providing details on which part of the respective workflow is supported by what kind of AI method.

1.3 Structure of the document

The rest of the document is organized as follows. Chapter 2 reviews the status of HiDALGO requirements in the 3rd year of the project. Chapter 3 provides details on the seven AI components as well as the AI-enabled use case workflows and Chapter 4 concludes with an overview of HiDALGO's data and component integration process. This document ends with three Annexes which contain information of not (yet) published work and link to the respective PDF documents. In addition, the latest lists of requirements and scenarios are attached as supplementary Annexes (as PDF documents).

2 Requirements Evolution

Section 2.1 reports on the status of the project's requirements at M36 of the project's lifetime (which was extended by 3 months), mainly focusing on results from the 3rd update iteration of requirements. Section 2.2 connects the requirements and scenarios to the project's objectives, which directly link to the project's KPIs.

2.1 Status of Requirements and Scenarios

The provided list of requirements and scenarios (initially captured in D6.1 [12] and updated in D6.4 [15] and D6.5 [16]) represents an understanding of what needs to be accomplished to conclude the project successfully. The understanding is open to change due to experiences made throughout the project and a constant learning curve.

Since it proofed beneficial in previous iterations, project partners were asked to review the requirements they were responsible for, to report changes, and to adapt them accordingly. The 3rd update iteration of requirements was conducted from May to June 2021.

In the following, we report the status of the requirements as of M36; since the project has been extended to February 2022 (M39), there are still ongoing requirements to report.

Altogether, six extra scenarios were added (SCO-PIL-012, SCO-PIL-013, SCO-PIL-014, SCO-POR-015, SCO-POR-016, SCO-POR-017) resulting in 49 scenarios, compared to 43 at the beginning of the project. The additional scenarios reflect larger changes in the project including the addition of a fourth – the epidemiology – use case, and the increased efforts regarding our training as requested in the mid-term review. 28 requirements were added (REQ-HPII-042, REQ-PIL-038, REQ-PIL-039, REQ-PIL-040, REQ-PIL-041, REQ-PIL-042, REQ-PIL-043, REQ-POR-047, REQ-POR-048, REQ-POR-049, REQ-POR-050, REQ-POR-051, REQ-POR-052, REQ-POR-053, REQ-POR-054, REQ-POR-055, REQ-POR-056, REQ-POR-057, REQ-POR-058, REQ-POR-059, REQ-POR-060, REQ-POR-061, REQ-POR-062, REQ-POR-063, REQ-POR-064, REQ-POR-065, REQ-POR-066, REQ-POR-067) resulting in a total of 162 requirements, in comparison to 134 at the start of the project.

The status of the requirements in M36 is as follows:

- 86 requirements are fulfilled. (53%)
- 54 requirements are ongoing, in progress, being optimised or partially done. (33%)
- For 22 requirements, it turned out that they are deprecated due to changes in the workflow. (14%)

These ongoing 54 requirements (33%) include:

- Requirements with lower priority including documentation activities such as REQ-PIL-038, REQ-POR-049 and REQ-AIS-004 and training activities such as REQ-POR-054.
- Requirements which are partly done such as REQ-PIL-020, REQ-POR-005, REQ-POR-015, REQ-POR-024 and REQ-POR-025 but still need refinement.
- Requirements which are already implemented but (using the opportunity of the extension) are being optimised such as REQ-PIL-006 and REQ-POR-015.
- Requirements related to visualization activities such as REQ-POR-039, REQ-POR-040 and REQ-AIS-008 which are often finalized at the end of a project's lifetime.
- Requirements related to finalizing the portal including testing, deployment and data management such as REQ-POR-022, REQ-POR-023, REQ-POR-024, REQ-POR-025, REQ-POR-026, REQ-POR-043, REQ-POR-044, REQ-POR-025, REQ-POR-061, REQ-POR-062.

As documented throughout the deliverables of this work package, the list of requirements may undergo changes, i.e., requirements are being added, adapted as well as deprecated, due to a constant learning process and experiences made. Table 1 provides an overview of selected requirements to exemplify changes during the 3rd year of the project.

Requirement	Description	Change Type	Explanation
REQ-HPII-042	Run Polytope-based scripts on ECMWF-cloud for data conversion	added	To provide data for SZE hydrostatic fluid solver.
REQ-POR-053	Provide HiDALGO-specific AI training by adapting the existing training curriculum to the specific needs of the consortium.	added	To adhere to reviewers' comments on intensifying our training activities.
REQ-HPII-020	Integrate simulation and data analytics tools to reach high performance on the HiDALGO infrastructure.	adapted	Added usage of Spack to combine it with Ansible configs for installing tools on the HiDALGO infrastructure.
REQ-POR-012	Implement yellow pages to serve as user guide	adapted	Postponed due to low priority of the task; even considered to be deprecated.
REQ-HPII-001	Multi-core and/or distributed implementation of simulators in SUMO package	deprecated	Focus has shifted to Python-based Flee ABMS framework.
REQ-POR-029	Provide training to users of ECMWF cloud how to use the cloud infrastructure.	deprecated	Interaction with the cloud is provided through the orchestrator.

Table 1. Selected requirements to exemplify and motivate addition, adaptation as well as deprecation.

The latest lists of requirements and scenarios are attached as supplementary Annexes (as PDF documents). These documents contain detailed information on the status of each requirement, i.e., (i) "Requirement done", (ii) "Requirement ongoing, in progress or being optimised" or (iii) "Requirement is deprecated".

2.2 Relation to Project Objectives and KPIs

This section connects the requirements and scenarios to the project's objectives (see Table 2), which directly link to the project's KPIs (see Table 3). As stated in D6.1 [12], collected user stories were formalized into scenarios, which in turn refer to specific requirements to be fulfilled. These scenarios were then grouped into four main categories, which are connected to the project's work packages:

- **HPC/HPDA Infrastructure and Implementation (HPII):** This category includes, e.g., improvements of simulation kernels, HPC-compliant frameworks for agent-based modelling, simulation, platform robustness as well as scalability aspects.
- **Pilots (PIL):** This category includes scalability aspects, simulation aspects concerning the pilot use cases and the important topic of coupling.
- **Portal (POR):** This category includes the development of the portal, including visualization methods, GUIs, support and documentation. In addition, it includes data management aspects such as security and authentication as well as ethical considerations such as protection of personal data.
- **Artificial Intelligence Support (AIS):** This category includes all aspects to best support the use case workflows by AI methods.

Capturing and keeping track of requirements is a pre-requisite for meeting the project's objectives and in consequence the project's KPIs. Deliverable D6.5 [16] provided an overview whether work packages, categories and main project objectives are appropriately connected by requirements. For the sake of completeness, this overview is illustrated in Table 2 updated by year 3 activities. There is only a minor change compared to D6.5 due to the adding of scenario SCO-POR-017 which explicitly addresses sustainability and thus contributes to BO2 "Develop and implement a roadmap for sustainable HiDALGO operation".

	HPII	PIL	POR	AIS
WP2	X		X	
WP3	X	X	X	X
WP4	X	X	x	
WP5	X		X	X
WP6	X		x	X
WP7			X	
WP8	X		X	

	HPII	PIL	POR	AIS
Project Objectives	SO 1,2,3 TO 1,2,3,4 CO 1,3,4 BO 1,2	SO 1,2,3,4 TO 1,2	SO 1,3 TO 3,4 CO 1,2,3,4 BO 1,2	SO 1,2,3,4 TO 1,2,3,4

Table 2. Relation between work packages, categories and overall project objectives, where SO stands for scientific objective, TO for technological objective, CO for community objective, and BO for business objective. A “X” denotes that there is a very strong connection of this category’s scenarios to the respective work package, a “X” denotes a strong connection and a “x” denotes that there is a connection.

This overview allows us to observe that (i) the POR category nicely connects all work packages – it appears to be the glue between them and that (ii) almost all work packages support the HPII category harbouring the important scalability aspects. This overview has thus served us as additional assurance that all the project’s objectives are taken care of appropriately.

The project objectives directly link to the project KPIs, i.e., taking care of the objectives has a direct impact to taking care of the KPIs of the project. We refer to the Grant Agreement [18] for a more detailed description of the project’s objectives (pp. 5 - 7) as well as KPIs (pp. 30 - 31). Table 3 links the project’s 16 KPIs to the respective scientific (SO), technological (TO), community (CO) and/or business (BO) objectives and to the respective work package deliverables for more details. Finally, D1.4 [1] will provide a critical analysis from the viewpoint of the technical management including the specific KPI target values.

Impact	KPI Description	Corresponding Objective(s)	Deliverable(s)
1	Number of new coupled simulations enabled Scalability reached with coupled simulations Future scalability for the pilots Number of new/adapted tools/applications	SO2, TO2 SO1, SO2, SO3, TO1, TO2 SO1, TO1, TO2 SO1, TO1	D3.5, D4.4, D5.8
2	Number of new/adapted tools/applications Improvement in the time to run simulations Number of new coupled simulations enabled Improvement in simulations accuracy due to coupling	SO3, TO3 SO1, TO1, SO2, TO2 SO2, TO4	D3.5, D4.4
3	Number of services provided by the CoE Number of companies reached Number of European companies participating in a HiDALGO demo	TO3, BO1, BO2 CO3, BO1 SO4, TO3, CO3	D2.4, D5.7
	Number of scientific domains involved	SO3, CO1, CO2, CO3, CO4, BO2	

Impact	KPI Description	Corresponding Objective(s)	Deliverable(s)
4	Number of domains involved in coupled simulations	SO2, SO4, TO2, CO4, BO2	D4.4, D7.5
	Number of federated resources	TO3, CO2, CO3, CO4, BO2	
	Number of trained scientists	CO2	D7.5
5	Number of scientists reached	CO1, CO3, BO1, BO2	

Table 3. Relation of KPIs to Project Objectives and respective Deliverables.

3 Artificial Intelligence enabled Workflows

In this chapter, we present the final versions of the use case workflows, with a focus on AI components. As discussed in D6.3 [14] and illustrated in Figure 1, we planned AI support at three different levels:

1. model creation and parametrization, e.g., pre-processing, feature engineering, etc. (left)
2. model execution, for instance, model-order reduction (middle), and
3. advanced result visualization and verification (right).

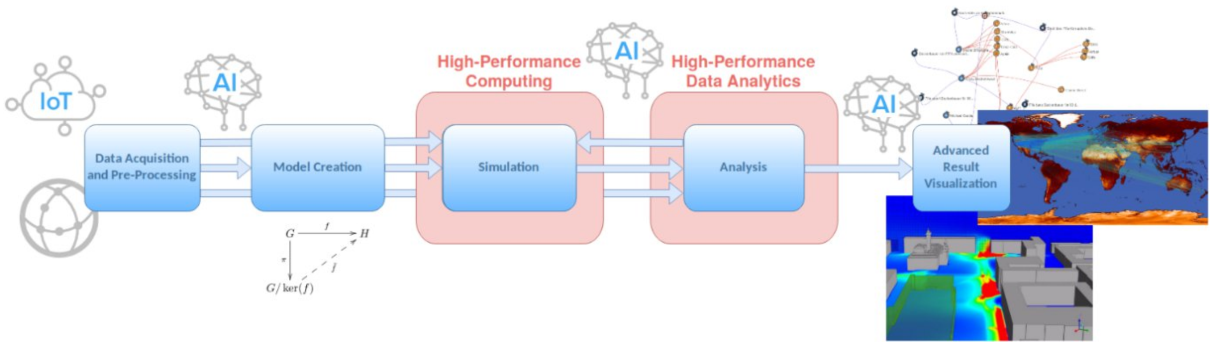

IoT
AI
Data Acquisition and Pre-Processing
Model Creation
High-Performance Computing
Simulation
AI
High-Performance Data Analytics
Analysis
AI
Advanced Result Visualization

Figure 1: Supporting the HiDALGO Workflow with AI at three different levels.

Since this report is final, its content should be self-contained. Thus, in contrast to intermediate reports, which describe only the incremental progress, some degree of redundancy with previous deliverables is inherent. Specifically, the workflow descriptions will largely overlap with D6.2 [13], while the description of the AI components overlaps with D6.4 [15] and D6.5 [16]. We nevertheless aimed at a concise description and thus defer details to corresponding publications (see [22], [26], [28], [29]). For the sake of completeness, we also briefly report work on discontinued components (and explain why they were discontinued). In the workflows, these are indicated as empty boxes.

To summarize, within HiDALGO we support three of the four use cases via seven AI components:

- Migration Use Case:
  - Location Graph Extraction (Section 3.1.1)
  - Study of Environmental Factors (Section 3.1.2; ongoing)
- Urban Air Pollution Use Case:
  - Proper Orthogonal Decomposition (Section 3.2.1)
  - Traffic Forecasting (Section 3.2.2; ongoing)

- Social Networks Use Case:
  - Community Detection (Section 3.3.1)
  - Synthetic Graph Generation (Section 3.3.2)
  - Synthetic Message Generation (in Section 3.3.3)

The fourth, Epidemiology Use Case, did not receive any AI support, because it was initiated at a stage in the project in which all resources for AI support had already been allocated. We nevertheless list this use case workflow for the sake of completeness in Section 3.4.

We further performed research towards several other AI tasks that are connected with HiDALGO's use cases, such as Markov aggregation for reducing agent-based models (Section 3.1.3), correlation analyses of traffic and air pollution data (Section 3.2.3), or investigations of selected Twitter corpora (Section 3.3.4). Finally, regarding advanced result visualization and visual analytics, we have investigated the use of Visualizer in the Migration and Social Networks use cases. The Visualizer was introduced and its application in the use cases was discussed in D3.4 [4]. Details about its integration can be found in D5.6 [9], with additional information in the concurrently submitted deliverable D5.7 [10]. The AI capabilities of Visualizer are introduced in D3.5 [5].

Before delving into the developed AI components, a clarifying comment regarding our usage of the term AI is in order. Artificial Intelligence can loosely be split into two categories: symbolic, rule-based AI (such as expert systems and logic) and sub-symbolic AI (such as machine learning and neural networks). While it has become common practice to equate AI with deep learning (which is undoubtedly one of the currently biggest subfields of AI), within HiDALGO we embrace the wider perspective of the original definition AI permits. In other words, some of our AI components are purely rule-based (such as the Location Graph Extraction component of the Migration Use Case), some are statistical (such as Proper Orthogonal Decomposition in the Urban Air Pollution Use Case), and some are based on neural networks (such as the Synthetic Message Generation Model in the Social Networks Use Case). As such, AI support in HiDALGO shall be understood as components (or parts of components) that solve some well-defined tasks using a set of rules, which are either learned from data or crafted manually. We are convinced that this wider perspective allows for a better support of the use cases than the more narrow, contemporary focus on neural networks would. To showcase the achievements of the developed components and their impact on the use cases more clearly, we provide a short summary in Section 3.5.

3.1 Migration Use Case

The final version of the workflow is visualized in Figure 2. AI support is provided at three different levels: (i) model creation and parametrization, for instance, pre-processing, feature engineering, etc. (visualized as red components), (ii) model execution, for instance, model-order reduction (visualized in blue), and (iii) advanced result visualization and verification (visualized in green). Regular, i.e., non-AI components are visualized in grey. Components that are currently under development and that are planned to be finished by M39 are indicated by dashed lines. Discontinued components are indicated as empty boxes.

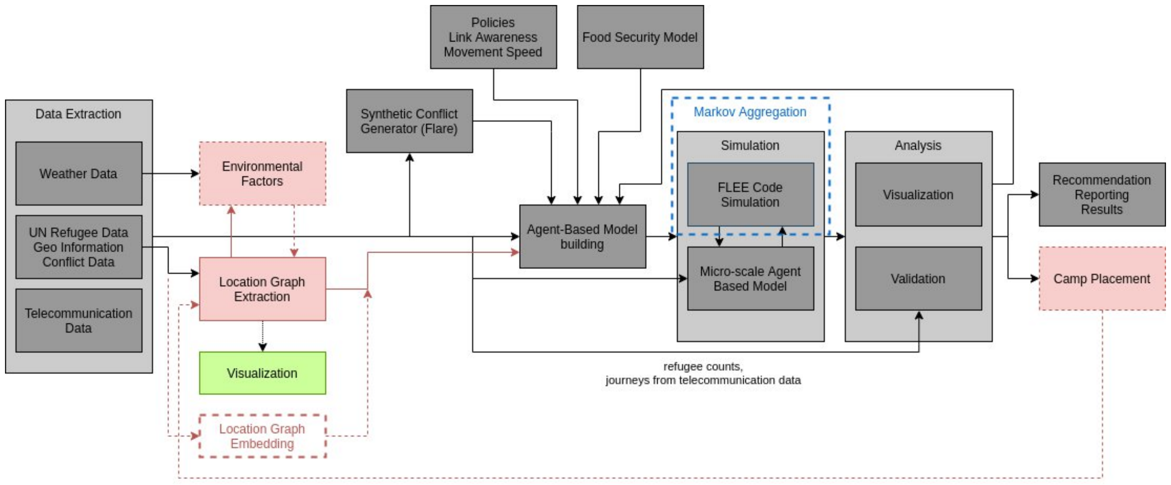


Figure 2: Final Workflow for the Migration Use Case.

The use case considers refugee data from the UN, conflict data, weather data, and geographical information about the conflict regions as *Data Sources*. From these sources, data is extracted for model building and validation. Map data and camp locations are used to create and visualize *Location Graphs*, which contain locations of interest and travel distances between them (Section 3.1.1); the *influence of weather* on travel distances is modelled separately (Section 3.1.2). *Location Graph Embedding* was supposed to speed up the creation of location graphs, but was found to be unnecessary and thus discontinued (Section 3.1.3). *Synthetic Data* enters the use case in the shape of policies, such as camp closures or re-routing of refugees, as well as agent parameters, such as movement speed and link awareness. Furthermore, the *Synthetic Conflict Generator (Flare)* creates a random sample of realistic conflict evolutions (including new outbreaks) based on recent conflict data from various sources. Under the assumption of evolving conflicts, this component enables forecasting of refugee movements and to prepare policy makers for reasonable eventualities.

All extracted and synthetically generated data is used for *Agent-Based Model building* (e.g., refugee counts determine the size of the agent population; policies, routes, and camp locations determine the agent environment; link awareness and movement speed determine agent rulesets). As part of this step, we also need to rely on specifications of other parameters (such as refugee movement speed), policy decision types that the user wishes to investigate, and relevant food security information or predictions derived from the *Food Security Model*. New models may also be constructed based on patterns identified in previous simulations.

The simulation itself involves the main instance of *Flee*, which simulates refugees escaping to neighbouring countries, as well as several *microscale instances*, which model escape from individual settlements. Initial ideas to replace Flee with a reduced-order *Markov model* were abandoned after improvements of the Flee code (Section 3.1.3). The output of each simulation is then prepared for analysis. The results are *visualized* and *validated* against the true refugee count per camp published by the UNHCR. Key findings will then lead to further recommendations.

3.1.1 Location Graph Extraction and Visualization

An essential input to the Flee code is a graph structure with physical locations as nodes and routes between the locations as edges. Additionally, the route distance between nodes is used as meta-information. The manual creation of such a graph is very time-consuming, error-prone and thus limited to roughly 50 locations. With a more advanced agent rule set, which would significantly improve simulation accuracy, location graphs with 500-500,000 vertices are needed. The AI component *Location Graph Extraction* thus

(1) automatically extracts a list of relevant locations (e.g., cities, towns, etc.) and their population either within a country or within a rectangular region via API calls to OpenStreetMap [32],

(2) creates a fully connected graph with (walking or) driving distances between the locations with the Open Source Routing Machine [33] operating on offline map data,

(3) prunes routes based on the triangle inequality and a pruning factor $\beta < 1$ to remove indirect connections between locations from the fully connected location graph, and

(4) visualizes the resulting location graph of the extracted region (using the OSMnx library [34]).

The route pruning algorithm has been applied to and evaluated in four geographical regions with several numbers of locations and various values for the pruning parameter $\beta \in [0.8, 0.99]$. The result is a location graph (see Figure 3) with direct driving connections between the locations that is used as input for the Flee code. The approach and its evaluation are published in [26] and code is openly accessible [35]. In these evaluations, our approach simultaneously achieved precision and recall scores exceeding 0.9 compared with a manually created ground truth. Furthermore, it was shown to scale well to locations graphs with larger numbers of locations; e.g., creating a location graph connecting 18.000 European cities was accomplished in a little over 5 hours on a single HPC node.

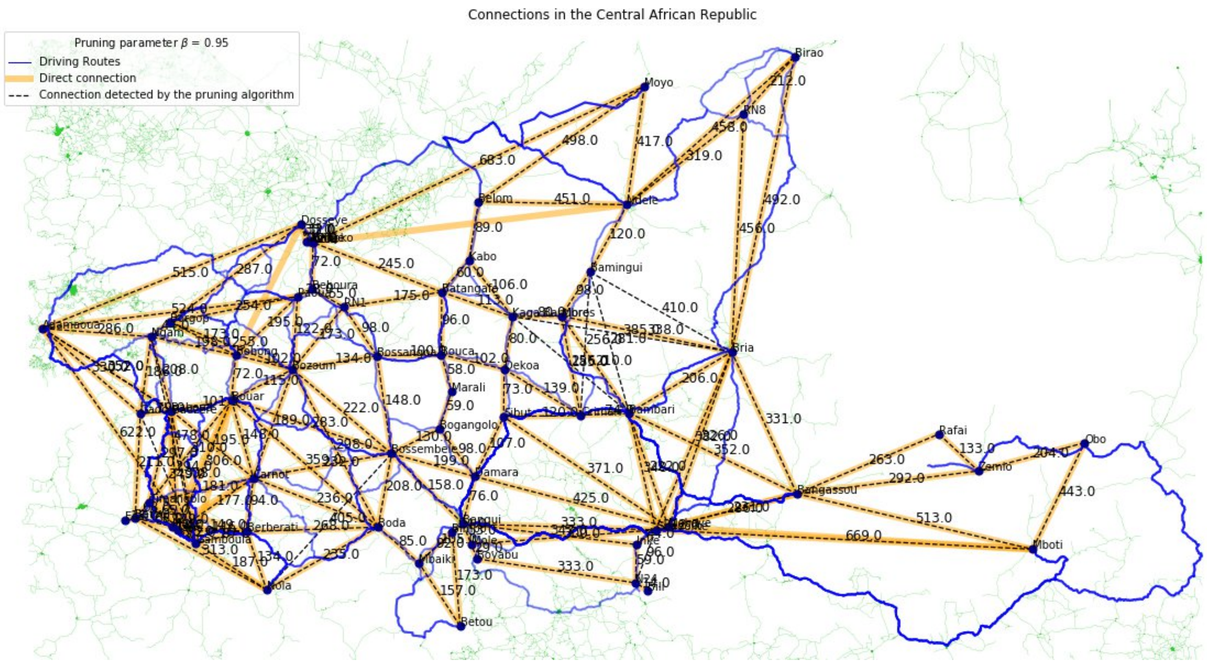


Figure 3: Connections of 62 locations in the Central African Republic with $\beta = 0.95$.

3.1.2 Environmental Factors Influencing the Location Graph

We analysed historical data about route distances, precipitation, and wind speed in the regions of interest. This analysis allows us to gain insight into how agents choose routes depending on season of the year, as routes become longer, shorter, or even unavailable. Most of the analysis is focused on South Sudan, a region with clear dry and wet season separation, where there is statistically significant correlation between historic route distances and weather conditions: Figure 4 shows that the majority of routes between pairs of locations is higher than average during wet season.

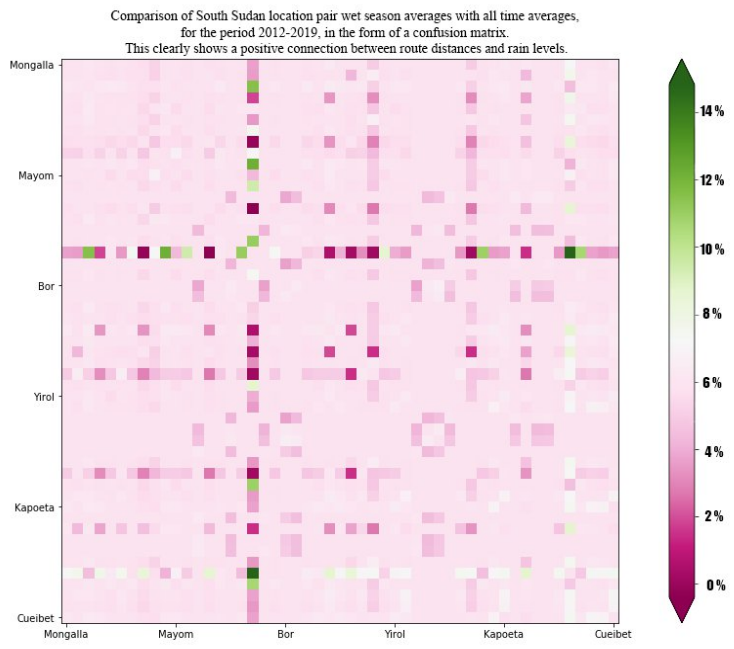


Figure 4: Effect of precipitation on route distances.

In order to model this correlation, we utilized a regression model that incorporates the precipitation values across the path that connects two locations. Learning from the period 2012-2015, we could extrapolate these values to predict route distances, given precipitation, for the following years with a margin of error of about 20%.

Future work will extend this analysis with the goal to incorporate the connection of weather conditions and route distances into the microscale models, where each agent will decide their journey based on weather conditions.

3.1.3 Other Contributions

Location Graph Embedding. The aim was to simplify route distance calculation by embedding locations in a high-dimensional Euclidean space such that the route distances are equal to the Euclidean distances between the high-dimensional representations of the locations. As the performance of the Location Graph Extraction code was shown to scale well with the number of locations, this research was discontinued.

Markov Aggregation. In the Migration Use Case, refugee movement is simulated using an agent-based model (ABM), which can be interpreted as a Markov chain with a state space corresponding to all possible agent configurations. This ABM is used to simulate aggregate statistics for which the identities of the agents are immaterial. We were able to show that, under certain simplifying assumptions, the corresponding Markov chain is lumpable with respect to the aggregation function, i.e., the quantity of interest can be computed by drawing from a Markov chain with a significantly smaller state space (see Annex A). This could (i) reduce the computational complexity of obtaining the results without affecting their accuracy and (ii) potentially make the computational complexity of obtaining the results independent of the number of agents. (The computational complexity of the current Flee code is at least linear in the number of agents.)

Since considerable performance improvements have been made in the implementation of Flee, we have discontinued this research direction. However, theoretical work that extends the concept of lumpability from Markov chains to Markov random fields has been published in [22].

Camp Placement as Multi-Objective Optimization Problem. For forced displacement, one of the most significant issues faced by the governments in the destination countries is camp placement. Camps are important infrastructures to provide protection and allocate available humanitarian resources to thousands of people being forced to flee from their hometowns.

Camp placement in the context of forced migration is especially challenging, which can be formulated as a multi-objective optimization problem (MOP), simultaneously minimizing individual refugee travel distance and camp idle capacity and maximizing the demand satisfied by the candidate camp. In addition, governments of receiving countries may need to minimize external costs (e.g., the cost of facility construction, the transportation of humanitarian resources). So, we formulated a simulation-based multi-objective model, where the Flee agent-based simulation is utilized to predict the movement of displaced people to a potential camp. The proposed multi-objective model involves the following components:

- Decision variables: the longitude and latitude coordinates of the camp.
- Objective functions: (1) the minimisation of individual travel distance, namely, the average distance travelled by each arriving agent in a destination camp at the end of the simulation should be minimised; (2) the maximisation of the demand covered by the camp, namely, the number of people in the camp at the end of the simulation should be maximized; (3) the minimisation of the amount of idle camp capacity, namely, the average remaining camp capacity over simulation days for the camp should be minimised.
- Constraints: the search space of the camp location is a square area outside the source country.

The optimisation of this multi-objective model is very complex due to the uncertainty of people's movement and high computational requirements for estimating the objectives of all possible camp locations. We thus employ evolutionary algorithms to search for the optimal camp locations that satisfy these objectives and constraints, and the Flee agent-based simulation is adopted to estimate the three objectives in the evolutionary process.

Evolutionary algorithms are a type of population-based metaheuristic algorithms, which are well suited to the complex multi-objective optimisation problem. The population-based search property of evolutionary algorithms could provide a set of optimal solutions in a single run of the algorithm, where each solution represents a trade-off among these objectives. Figure 5 illustrates our proposed optimisation approach.

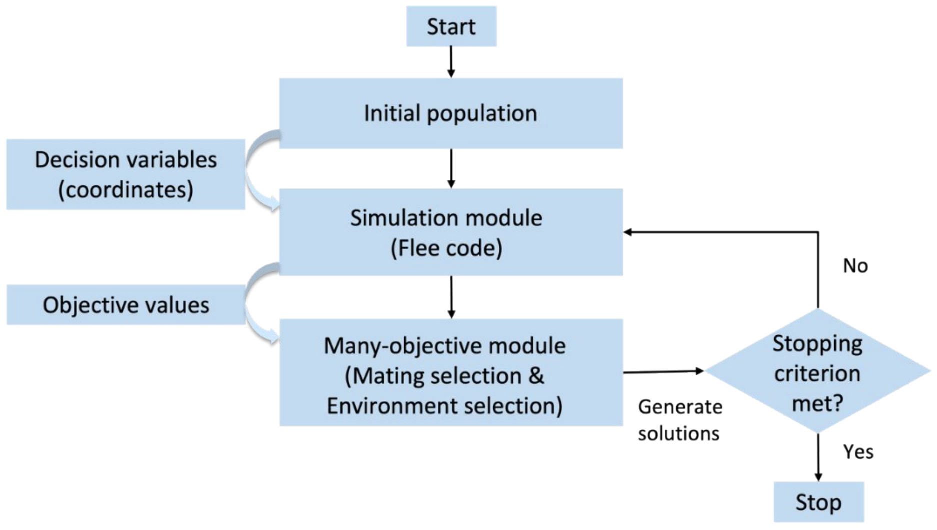

Start
Initial population
Decision variables (coordinates)
Simulation module (Flee code)
Objective values
Many-objective module (Mating selection & Environment selection)
Generate solutions
Stopping criterion met?
No
Yes
Stop

Figure 5: Flowchart of the optimisation technique for camp placement.

First, a population (or a set of solutions), i.e., a set of possible camp locations are randomly generated. Then, these locations (or coordinates) are used as the input parameters for the simulation module. By running the simulation module, a set of corresponding objective values are obtained for the current population. Next, the multi-objective module is performed to generate an offspring population. The offspring population will then be evaluated by the simulation module. This process will continue until a stopping criterion is satisfied. Note that the multi-objective module includes two important evolutionary operators (i.e., mating selection and environment selection), which can be implemented flexibly. For further details and initial results, we refer the reader to D4.4 [8].

3.2 Urban Air Pollution Use Case

The final version of the workflow is visualized in Figure 6 below. AI support is provided at three different levels: (i) model creation and parametrization, for instance, pre-processing, feature engineering, etc. (visualized as red components), (ii) model execution, for instance, model-order reduction (visualized in blue), and (iii) advanced result visualization and verification (visualized in green). Regular, i.e., non-AI components are visualized in grey. Components that are currently under development and that are planned to be finished by M39 are indicated by dashed lines.

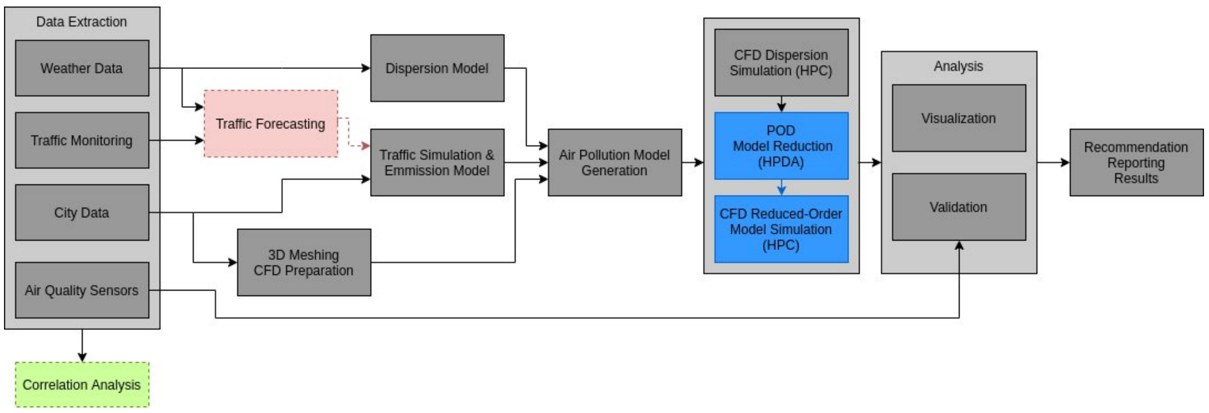

Data Extraction
Weather Data
Traffic Monitoring
City Data
Air Quality Sensors
Correlation Analysis
Traffic Forecasting
Dispersion Model
Traffic Simulation & Emmission Model
3D Meshing CFD Preparation
Air Pollution Model Generation
CFD Dispersion Simulation (HPC)
POD Model Reduction (HPDA)
CFD Reduced-Order Model Simulation (HPC)
Analysis
Visualization
Validation
Recommendation Reporting Results

Figure 6: Final Workflow for the Urban Air Pollution Use Case.

The *Data Sources* contain meteorological data, historical traffic data, traffic data from monitoring loops and traffic cameras, air quality sensors and global air quality services like CAMS simulations, road information from OpenStreetMap, and geospatial information about the city. After *Data Extraction*, some of this data will undergo a *correlation analysis* or be used to *generate synthetic data*: Geospatial city information will be used to create a 3D model of the city; road information and traffic information will be used to run traffic simulations (using SUMO and a COPERT emission model) on both macroscopic and microscopic scales; and historical traffic data and weather forecasts will be used to synthesize *traffic forecasts* (Section 3.2.2).

External models, pre-processed weather data, traffic simulations, an advection-diffusion model, and the 3D city model are combined to generate an *Air Pollution Model*, which is simulated by running 3D CFD solvers with time-varying boundary conditions (changing traffic and weather) – this is represented by the block *CFD Dispersion Simulation*. We furthermore execute *Proper Orthogonal Decomposition (POD)* (Section 3.2.1) after collecting sufficiently many snapshots, which allows us to perform *Reduced-Order Model Simulations*. The results are *visualized* and *validated*, for example against measured NOx concentrations.

3.2.1 Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) ([30], [20]) is a method for reducing the order of complexity of simulating systems of differential equations. We applied this method to numerical solutions of time-dependent problems of gas dynamics: numerical solution of the compressible Euler and the compressible Navier-Stokes equation, as in the computation of the wind in cities. Just like principal components analysis or singular value decomposition, POD is a data-driven technique in the sense that the reduced-order model is based on the simulation results of the full-order model. Thus, integrating POD into the workflow requires two phases: an offline phase, in which several full-order simulations are run to collect a snapshot matrix from which the basis vectors of a reduced-order basis are computed, and an online phase in which the differential equations are only solved in this reduced-order basis. Thus, for a given scenario, either the full-order model or, if a snapshot matrix has already been created, the reduced-order model is simulated.

The accuracy of the POD method depends highly on the quality of the geometry of the simulated domain and its mesh. It was observed that in each use-case (simulation for the city of Győr and Antwerp) the requested 0.01 relative accuracy is achieved at dimensions as small as 25, and, surprisingly, the simulations were stable at huge step sizes (with CFL-number up to 1000), which is a significant scientific result; its publication is ongoing and will be submitted within the project life-time. The running time of POD dropped to 1/70 of the running time of the full order model on the same hardware infrastructures (CPU compute nodes and NVIDIA A100 GPUs). For more details on this POD method, we refer to D4.4 [8].

Note that POD trades accuracy of the solution with the computation time required to get it. Thus, the parameters of POD need to be set differently for each scenario, adapting to the required accuracy.

3.2.2 Traffic Forecasting

The traffic forecasting component creates synthetic traffic measurements, such as those obtained from traffic loops or traffic cameras, compatible with SUMO. This synthesis relies on historical data and models the traffic at a measurement point as a cyclo-stationary stochastic process with a period of 168 hours. In other words, the component computes the forecast for traffic loop/camera data for a given time window by averaging corresponding time windows within the historical records. This approach is inherently robust to missing values and robustness to outliers in historical records is achieved via outlier detection (median absolute deviation). Trends in traffic data are accounted for by adjusting the time window in which the averages are taken. The module is capable of delivering traffic forecasts at arbitrary intervals, and initial experiments suggest that the forecasts achieve a coefficient of determination (R2) exceeding 0.7 or even 0.8 for intersections with strong traffic (the accuracy was worse for sections with little traffic, where regular patterns are less clear). A series of representative forecasts is shown in Figure 7, which shows that the weekly (less traffic on the weekends) and hourly (e.g., morning and afternoon rush hours) are captured well.

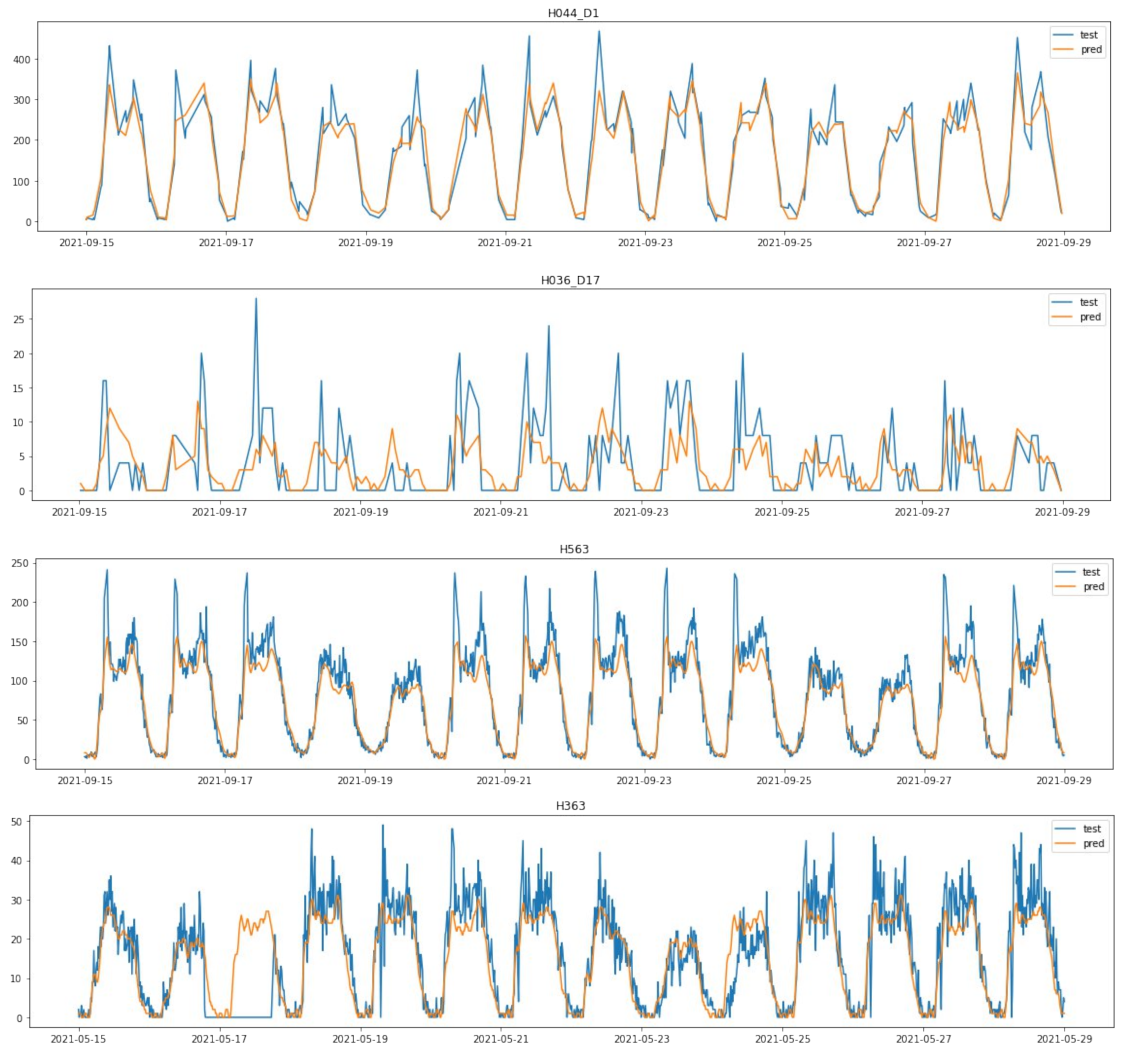


Figure 7: Traffic forecasts based on two months of traffic loop (top two) and traffic camera (bottom two) data, with R2 scores of 0.91, 0.29, 0.87, and 0.67 (top to bottom). It can be seen that weekly and hourly patterns match well, except in cases where there is generally little traffic (second image from the top). Also, camera data seems to yield more accurate forecasts.

In the remaining duration of the project, we will convert the current prototype to a script-based release and perform an in-depth evaluation to assess the accuracy of traffic forecasting.

3.2.3 Other Contributions

Correlation Analysis. We studied the connection between traffic density and pollution measurements coming from a number of weather stations. Using the aggregated intersection data, we constructed a traffic - pollution time series that can be used to create models and test them on their ability to predict future pollution values. We used several regression models, including Facebook's Prophet tool [27]. Of these models, Auto-ARIMA (Auto Regressive Integrated Moving Average, [31]) performed best (see Table 4).

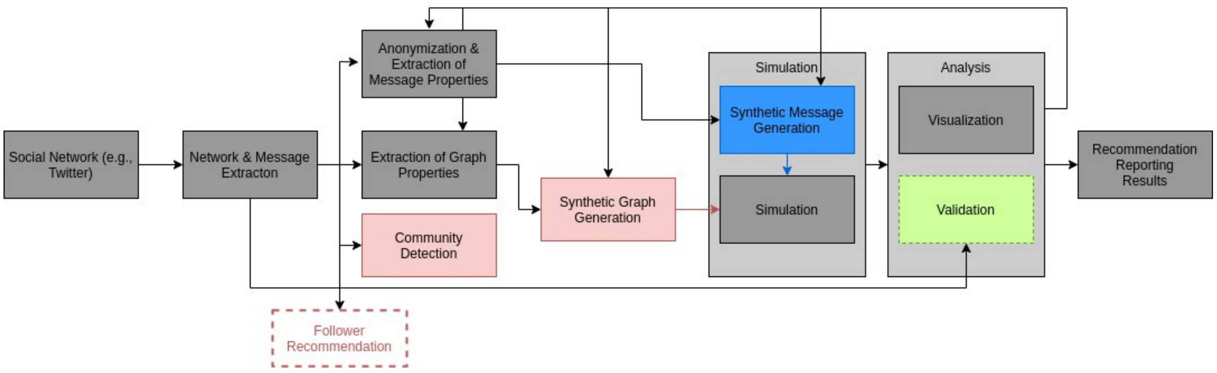
Model	MAE
Prophet	15.14
Prophet - w/ Logistic growth	15.97
Support Vector Regression (SVR)	13.90
Huber Regressor	14.32
Auto-ARIMA	12.77

Table 4. Prediction accuracy of air pollution for the following month based on four months of historical data.

A bivariate approach has been applied on the logistic growth Prophet model, while other univariate models provided better results. Indicatively, the seasonality coefficient of Auto-ARIMA that has provided the best results was calibrated to 24, which is the time window in hours that expresses more accurately the seasonal periodicity of data.

3.3 Social Networks Use Case

The final version of the workflow is visualized in Figure 8 below. AI support is provided at three different levels: (i) model creation and parametrization, for instance, pre-processing, feature engineering, etc. (visualized as red components), (ii) model execution, for instance, model-order reduction (visualized in blue), and (iii) advanced result visualization and verification (visualized in green). Regular, i.e., non-AI components are visualized in grey. Components that are currently under development and that are planned to be finished by M39 are indicated by dashed lines. Discontinued components are indicated as empty boxes.



Social Network (e.g., Twitter)
Network & Message Extracton
Anonymization & Extraction of Message Properties
Extraction of Graph Properties
Community Detection
Synthetic Graph Generation
Simulation
Synthetic Message Generation
Simulation
Analysis
Visualization
Validation
Recommendation Reporting Results
Follower Recommendation

Figure 8: Final Workflow for the Social Networks Use Case.

We consider social networks such as Twitter and Facebook as _Data Sources_. After extraction of these data, three components process them further. One of these components anonymizes messages and _extracts relevant message properties_, while the other component _extracts graph properties_ from the crawled social network data. A third component performs _Community Detection_ (Section 3.3.1). The extracted graph properties are then used for _Synthetic Graph Generation_ (Section 3.3.2). On the resulting graph, _Synthetic Messages_ (Section 3.3.3) are created and distributed, which is the heart of the _Simulation_. Both, synthetic graphs and synthetic messages, should have similar properties as the original data. The simulation in turn yields estimates for the spread of a particular message in a given network. These estimates will be _visualized_ (e.g., via summary statistics) and _validated_ (Section 3.3.4) against message diffusion recorded from real networks. The validation will be used to fine-tune the extraction of relevant features and model generation, while the obtained results may be used to issue _recommendations_ (e.g., regarding the detection of fake news).

3.3.1 Community Detection

Community detection supports the Social Networks Use Case in terms of model building and simulation validation: Investigating the community structure of a social network is necessary for adequate creation of synthetic graphs, as well as for evaluating the properties of the created graphs.

The AI component *Community Detection* is based on information theory and takes inspiration from the famous Infomap method [25]. While Infomap formulates community detection from the perspective of the minimum description length principle, our method Synwalk takes the perspective of random number generation and is closely related to the aggregation of Markov chains. We have carefully tested and evaluated Synwalk both theoretically and experimentally on synthetic and real-world datasets, indicating competitive performance in practically relevant scenarios. The approach is accepted for publication at ECML 2022 (journal track, see [28] for a preprint) and its code is freely accessible [36]. Extensive benchmarking of the code has been performed in WP3, with results being reported in D3.5 [5].

We also developed a tool for detecting overlapping communities, which first clusters the neighbourhood of each node using the Louvain method. Next, it removes communities according to some optimization function, e.g., clusters with small node or edge count or those contained in other clusters are removed. The remaining clusters are merged as follows: if the second largest eigenvalue of a cluster we would obtain by merging two randomly chosen overlapping clusters is higher than the second largest eigenvalues of the two unmerged clusters, then we combine the two clusters to one. This procedure ends in a local optimum with respect to the eigenvalues of the resulting clusters.

3.3.2 Synthetic Graph Generation

To simulate the spreading of messages, we developed a synthetic graph generation method that creates simple directed graphs involving reciprocal edges (two edges between two nodes with opposite direction) that have a predefined number of nodes and exhibit similar topological and algorithmic properties as real-world social networks.

We crawled several graphs from Twitter and used these real-world networks to tune the hyperparameters of our method: We fit χ^2-distributions to the reciprocal, in- and out-degrees of the nodes; generated correlated node degrees from the fitted distributions using copulas; placed edges between nodes with an approach similar to the Chung-Lu model [21]; and increased the average clustering coefficient by a variation of Bansal's edge rewiring algorithm [19].

Indeed, our approach turned out to synthesize graphs with similar topological (e.g., node degree distributions, density, sizes of the largest connected components, average clustering coefficient) and algorithmic (e.g., push-pull algorithm, epidemic spreading in the SIR model) properties. A paper describing our approach in greater detail and providing an in-depth evaluation is currently under review; a confidential copy can be found in Annex B.

We also designed a generator for undirected synthetic graphs and compared their structural properties to those of the corresponding real-world social networks.

The graph generation algorithm is based on the concept of overlapping communities. In the real world, people belong to multiple communities and social ties are established through one or more of these communities. We computed with a sophisticated clustering method these communities (see Section 3.3.1) and then analysed them with respect to the distribution of their sizes and the node degrees therein. These properties were finally used to create artificial communities. For each of the artificial communities a sub-graph was generated using the so-called configuration model [24].

For the validation, we used several structural properties such as the degree distribution, clustering coefficient, or the average distances in the graph. Furthermore, we compared the eigenvalue histogram of the generated synthetic graphs to that of the corresponding real-world networks. Note that this graph generation method is highly scalable.

3.3.3 Synthetic Message Generation

A necessary ingredient for the simulation is the retweet probability for a tweet posted by a user. Via data-driven modelling and estimation of retweet probabilities, the synthetic message generation component is supported by AI.

To determine the most influential features on retweetability, five Twitter datasets (totalling >300k tweets) were purchased. We investigated 25 different candidate features in various combinations, from which we selected 8 binary features. The influence of these binary features was modelled using logistic regression and a fully connected feed-forward Neural Network (NN), respectively. The results returned by the logistic regression model and several tested NN architectures were compared and it showed that using a NN represents the relative retweet frequencies more accurately. A dense NN with 3 hidden layers and 16 nodes per hidden layer returned the best results. With this model, the retweet probability of a new tweet can be estimated and used inside the simulation framework. For a more detailed description of the various data enhancement, processing and modelling steps see Annex A1 of D6.5 [16].

The resulting probabilistic model is then combined with the covariance of the transmission probabilities between neighbouring edges and a discount factor for decreasing the probability with increasing distance from the author of a message.

3.3.4 Other Contributions

Validation of Social Networks Use Case. The model that simulates the propagation of tweets works on binary features, and uses metrics like retweet probability and mean retweet counts in order to inform how likely a tweet is to be retweeted. We have focused on assessing the ability of the simulator to preserve these metrics that are learned from the ground truth, when it simulates retweets.

```
Algorithm 1 Twitter simulation validation
Require: dataset $\mathfrak{D}$, number of sources $so$, number of samples $sa$
 1: procedure SIMULATION_VALIDATION($\mathfrak{D}, so, sa$)
 2:     calculate statistics $stats$ from $\mathfrak{D}$
 3:     use $\mathfrak{D}$ and $stats$ in order to create a simulation object $sim$
 4:     for every feature vector $f_i$, $i = 1, 2, \ldots, n$ do
 5:         extract true mean retweets $mr_i^T$ and retweet probabilities $rp_i^T$ from $stats$
 6:     end for
 7:     for every feature vector $f_i$, $i = 1, 2, \ldots, n$ do
 8:         calculate predicted mean retweets $mr_i^P$ and retweet probabilities $rp_i^P$ from $sim(f_i, so, sa)$
 9:     end for
10:     return (MAE($mr^T, mr^P$), MAE($rp^T, rp^P$))
11: end procedure
```

Figure 9: Algorithm for validating the social networks simulation.

To this end, we developed a validation scheme depicted in Figure 9. The first stage of experiments proved that, across all Twitter datasets, these aggregate metrics are indeed preserved within a reasonable margin of error (see Table 5).

Dataset	MAE for #retweets	MAE for retweet prob.
NEOS outer	3.47/3.39	0.0033/0.002
FPOE outer	3.71/3.72	0.0019/0.002
Vegan outer	0.59/0.60	0.0011/0.002
Schalke inner	11.13/30.6	0.0193/0.0186
BVB inner	6.08/25.8	0.0216/0.0235

Table 5. Accuracy of the social networks simulation as measured by the mean absolute error (MAE) of the number of retweets and probability of retweets.

The second stage of experiments has been focused on parameter tuning. The simulation runs are based on parameters like propagation decay factors and maximum tweet travel distance.

As the processes that are used to approximate these parameters are resource-intensive, we are currently studying the trade-off between parameter tuning and simulation effectiveness.

As part of the validation module, we initially planned to compare the graph-theoretic properties of synthesized retweet cascades with actual retweet cascades in the corpus. Inferring these actual retweet cascades is a non-trivial task. We were able to show that under certain assumptions this task can be greatly simplified and coincides with finding a spanning tree on a directed, but unweighted graph. This insight is accepted for publication at Complex Networks, and a preprint of these results is available in Annex C.

Message spread on Social Media. We investigated the spreading behaviour of tweets posted by verified users in Twitter discourse networks of two Austrian political parties, the NEOS and the FPOE, in the lead-up to and during the 2019 Austrian National Council election, focusing on the gender of the author and the coded sentiment of the message towards the party.

In the FPOE-related network we observed that there are more tweets with a negative sentiment towards the party, which have a higher retweet rate and get retweeted more often on average compared to tweets with a positive or a neutral sentiment. In the NEOS-related network, the majority of tweets have a positive sentiment towards the party. The retweet rate is similar for each sentiment, but the mean retweet count for tweets with a negative sentiment is much higher. Regarding the gender of the users, in both networks most tweets were posted by a neutral gender, e.g., a newspaper, but they have the lowest mean retweet count. Female users account for the least number of tweets in both networks, with the retweet rate being about the same as for male users. In the FPOE-related network, tweets posted by male users tend to get more retweets. Our findings are summarized in [29].

Follower Recommendation. We also investigated whether the social network – in this case, the graph of follower relationships – contains patterns that can be used to learn recommendations for follower relationships. To this end, we trained a PyTorch-BigGraph embedding of the follower graph. As the relevance for the use case was not clear, this development was discontinued after initial exploration.

3.4 Epidemiology Use Case

The final version of the workflow is visualized in Figure 10.

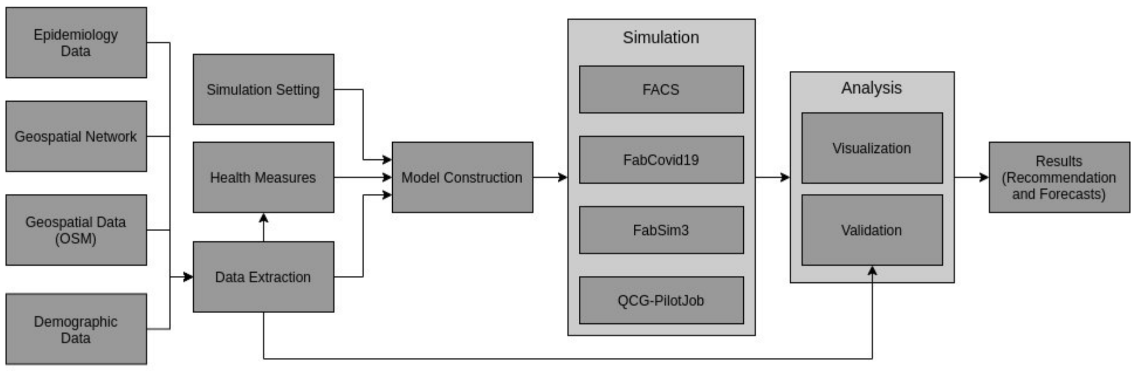


Figure 10: Final Workflow for the Epidemiology Use Case.

This workflow makes use of OpenStreetMap data, Epidemiology data, and demographic data as *Data Sources*. Data are *extracted* using built-in parsing tools; e.g., buildings and residential areas within a predefined region are extracted from OpenStreetMap. These data are used for *Model Construction*, taking into account different *Health Measures* (such as lockdowns, school closures, test and tracing, airborne transmission, etc.) as inputs for the execution of *Simulation* runs. Simulation is performed using the *FACS* component which models the spread of infectious disease at a local level and provides estimates of hospital arrivals for different scenarios. The *FabCOVID-19* and *FabSim3* modules are used for automating ensembles and replications of large-scale FACS simulations. The *Analysis* component is responsible for post-processing and visualization (e.g., SEIRD graphs) of the simulation results and model validation. These outputs are produced in the form of a COVID-19 forecast report for the selected region. A more detailed description of the workflow can be found in D4.3 [7].

This use case was introduced at a later stage during the project. At this stage, all resources regarding AI support had already been allocated. This use case thus does not contain any AI components.

3.5 Summary

In summary, within HiDALGO we developed seven AI components, performed research towards several more, and published five papers on this topic with further papers planned after the remaining components are finished. To make these achievements, we relied on the traditional definition of AI containing both machine learning and expert systems, i.e., admitting rule-based as well as learned algorithms and models. While we have already hinted at the impact of these AI components in the respective sections, the Table 6 provides an overview, listing the component and the year(s) in the project it was developed, the main algorithmic approach underlying it, and its impact on the respective use case workflow.

AI Component	Description	AI Category	Impact on Use Case
Migration Use Case			
Location Graph Extraction (Y1, Y2)	Computes route distances between given locations, prunes redundant routes, and creates a location graph	Rule-based AI	Enabling component; essential for creating large models
Environmental Factors (under development; Y2, Y3)	Estimates of environmental effects (e.g., precipitation) on route distances	Machine learning (regression analysis)	Improves accuracy of route distances (and thus of simulations) by coupling with weather data
Urban Air Pollution Use Case			
Traffic Forecasting (under development; Y3)	Forecasts traffic based on historical traffic data	Machine learning (regression-based time series forecasting)	Enabling component; essential for creating air pollution forecasts
Proper Orthogonal Decomposition (Y2, Y3)	Model order reduction for systems of differential equations	Statistical modelling (singular value decomposition)	Trades between accuracy and simulation run-time
Social Networks Use Case			
Community Detection (Y1, Y2)	Partitions social network graph into overlapping/non-overlapping groups of users	Machine learning (clustering)	Enabling component; enables data exploration and targeted synthetic graph generation
Synthetic Graph Generation (Y3)	Creates synthetic social network graphs with realistic properties	Statistical modelling, rule-based AI (generative modelling, rewiring, etc.)	Enabling component; essential for social networks use case
Synthetic Message Generation (Y1, Y2)	Estimates retweet probabilities as a function of tweet and user features	Deep learning (non-linear regression)	Enabling component; essential for social networks use case

Table 6. Summary of AI Components in HiDALGO.

4 HiDALGO Integration

Section 4.1 contains information about integration of static and streaming data sources into the respective use cases, i.e., Migration (MIG), Urban Air Pollution (UAP), Social Networks (SN) as well as Epidemiology (COV). Section 4.2 provides information on the integration of different HiDALGO components (with a year 3 focus) into a single solution.

4.1 Data Integration

4.1.1 Static Data

Table 7 summarizes information about integration of static datasets; most of them are integrated with CKAN. With the exception of traffic and air quality sensory data, the datasets are either directly uploaded to the CKAN or accessed via CKAN links to the external resources.

Data Set Name	Use Cases	Technology	Data Provider	Access Rights
Food security data	MIG	CKAN	https://www.ipcinfo.org/	open
Conflict data	MIG	CKAN	https://acleddata.com/	open
GIS (OpenStreetMap)	MIG, UAP, COV	CKAN	https://download.geofabrik.de/	open
Stanford Large Network Dataset Collection	SN	CKAN	https://snap.stanford.edu/data/	open
Telecommunication data (CDR)	MIG, SN	REST API	MOON	consortium
Traffic and air quality sensory data	UAP	CKAN	ARH, MK	consortium
Twitter	SN	CKAN	Twitter Inc.	open
UN conflict and refugee data	MIG	CKAN	https://www.unhcr.org/refugee-statistics/	open
Climate and environmental data	MIG, UAP	REST API	ECMWF Copernicus Climate data store	open
Weather and climate data	MIG, UAP	REST API	ECMWF	open

Table 7. Integration of Static Datasets.

For the Migration pilot, we have exposed the IPC food security data, OpenStreetMap data, and ACLED data to the application via CKAN. In all cases, the data undergoes a number of pre-processing steps before being exposed to the application via CKAN. For instance, in the case of ACLED data, the battles from the relevant regions and time periods are extracted and exported to CSV format.

Within the SN use case, we acquired Tweet datasets from Twitter. From the Tweets we extracted the user IDs and retrieved the related Twitter follower graphs. The Tweets and the graphs have been prepared to meet the data protection regulations. The anonymized IDs of the Tweets and of the graphs correspond, so that they can be processed together. Furthermore, from the anonymized Tweets the defined features are extracted and stored in separate files. The Tweet features, the graphs and also the anonymized Tweets are provided on CKAN. For the simulation, the data can be retrieved via blueprints or simply by download.

The *Pokec* social network graph [37], the most popular on-line social network in Slovakia, has been acquired from the Stanford Large Network Dataset Collection. It is provided on CKAN after some pre-processing steps. This dataset serves mostly as dataset for benchmarking and validation efforts.

Since traffic and air quality data may contain sensitive information, they are subject to license agreement. Being stored in the SZE instance of CKAN, the data can only be accessed by consortium members after signing non-disclosure agreements. The traffic data providing system contains about 85 cameras with a server appliance, which collects data in real-time from these cameras. Every day, data is collected from the previous day, a CSV file with the relevant data is generated and uploaded to CKAN. Two different datasets are maintained: one with SHA256 encoded plate numbers (GDPR), and one without. In addition, statistical data from the Hungarian Road Maintainer loop networks are collected as well and are uploaded to CKAN.

For further information about data integration via CKAN, we refer to the deliverables D3.3 [1] and D6.5 [16]. Along with CKAN, HiDALGO provides a customized RESTful API for accessing telecommunication data from MOON, as well as for accessing weather and climate data from ECMWF. Implementation of the latter is called WCDA which is described in more detail in D3.4 [4] and D6.5 [16]. Static coupling of climate and hydrological data using the data from the Climate Data Store RESTful API (CDS API) has been successfully demonstrated and described in the deliverable D4.2 [6].

Telecommunication data are provided statically to the HiDALGO partners via the CKAN portal. The “Sudan traffic” data set is described in D4.3 [7]; it contains 5,029,848 Call Detail Records (CDR) from June 2016 to May 2017. D3.4 [4] provides a discussion on the lack of use of the provided data set. However, it does not provide the spatio-temporal distribution of mobile phones of the anonymised users and could not be coupled with simulations of forced displaced refugees.

4.1.2 Streaming Data and Hybrid Approaches

Weather forecasts, climate re-analysis, global hydrological and air quality data produced by ECMWF and Copernicus play a vital role in coupling with the Migration and Urban Air Pollution use cases. The Weather and Climate Data API (WCDA) was developed to enable dynamic coupling of the real-time weather forecast via RESTful API and processing data on ECMWF's European Weather Cloud (EWCloud) using Cloudify orchestrator. For WCDA details, we refer to D3.4 [4] and D3.5 [5]. To facilitate the data processing on the EWCloud, the Cloudify Croupier plugin for orchestrating HiDALGO workflows was extended to support access to EWCloud resources, to perform data retrieval, and run additional post-processing steps. The processing algorithm and the overall workflow for the UAP use case have been described in D5.7 [10] and D4.4 [8].

Data Set Name	Use Cases	Technology	Data Provider	Access Rights
Traffic and air quality sensory data	UAP	CKAN	ARH, MK	consortium
Twitter	SN	CKAN + monitor	Twitter Inc.	open
Weather forecast data	UAP	REST API	ECMWF	consortium
Climate and environmental data	MIG, UAP	REST API	ECMWF Copernicus Climate data store	open

Table 8. Integration of Streaming Data and Hybrid Approaches.

Table 8 provides an overview of the integration of streaming data and hybrid approaches. In order to integrate Twitter data, PSNC implemented a Cloudify blueprint named Twitter-monitor that allows to create statistics of tweets and launch Social Network Simulator. The statistics from Twitter-monitor are uploaded to the CKAN. The statistics are the input to the Social Network Simulator. The Social Network Simulator blueprint is launched asynchronously from the Twitter-monitor.

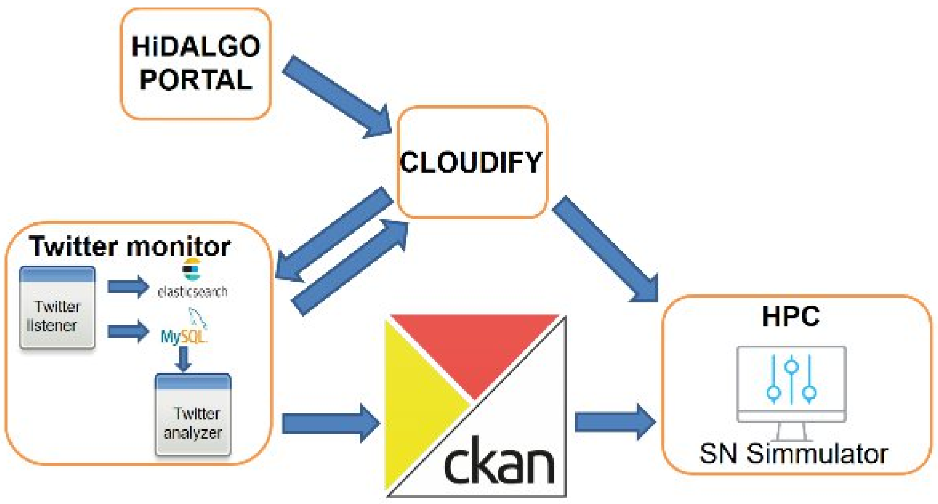


Figure 11: The workflow of Twitter-monitor, Cloudify and Social Network Simulator.

As illustrated in Figure 11, the Twitter-monitor will be run from the HiDALGO portal; if statistics are to be generated, the monitor will run the Social Network Simulator using Cloudify.

With respect to the traffic data, our partner Bosch uses the camera data and generates a SUMO simulation within its Cloud infrastructure, and uploads all the necessary input and config files to CKAN. There, special simulations can be run to analyse traffic and emission data.

Dynamic coupling of telecommunication data is done via a REST API. The API enables users to automatically request and retrieve available data sets via HTTPS from the MOON data servers for use in the Migration (MIG) and Social Networks (SN) use cases. The delivered CDRs are aggregated according to the MIG or SN use cases. The API delivers the requested data in plain CSV or zip-compressed CSV. Deliverable D3.5 [5] provides a complete description of the API.

4.2 Integration of Use Case Component Integration

This section covers aspects related to the integration of HiDALGO components focusing on year 3 developments. The chapter is split into two parts: Section 4.2.1 describes integration of the components which are runnable on HPC and HPDA infrastructures of HiDALGO, while Section 4.2.2 covers the integration of the HiDALGO services into an entire solution with the HiDALGO Portal as a single-entry point.

4.2.1 HPC, HPDA and AI Components of Pilots

HiDALGO supports three mechanisms for integrating software components of pilots: scientific workflows, automated deployment and software configuration management (AD/SCM), as well as continuous integration and continuous delivery (CI/CD). We strive to make the integration process uniform and easily reproducible. Scientific workflows for pilots are implemented by means of TOSCA blueprints which allow to run all use cases with Cloudify orchestrator in a similar manner. HiDALGO scientific workflows are discussed in detail in D6.5 [16]. In contrast, this section focuses on the automated deployment and CI/CD mechanisms for pilots. In their final shape, the latter mechanisms were developed in the last year of the project, and, thus, were only superficially covered in the previous deliverables. Below, we fill this gap shed the light on the current implementation of AD/SCM and CI/CD mechanisms of HiDALGO.

4.2.1.1 Automated Deployment

HiDALGO uses a combination of Ansible and Spack ([23], [38]) as a final solution for automated deployment. Compared to alternatives discussed in previous deliverables (e.g., containerization in D6.5 [16]), this approach allowed to achieve the following additional goals:

- support combinatorial versioning and reproducible software environments for all use cases over HiDALGO platforms and testbeds;
- customize builds consistently for each platform;
- facilitate separable software environments which reuse available software installations;
- support installation matrices for benchmarks and performance studies (see D5.8 [11]);
- enable off-line bare metal installation along with a full control on the installation process.

Spack is an HPC-oriented package manager for installing software packages from sources which allows to optimize the software to the target architectures and study effects of different configurations. Besides support for the management of dependencies, it resolves classic issues, typical for installation of software in HPC environments, such as combinatorial versioning, ban of Internet access for selected sites, accounting for site- and system-specific details (diversity of build systems, compilers, recommended setups, pre-installed software stacks). In addition, Spack supports separable software environments, which can be automatically converted to containerization recipes for Docker and Singularity. In order to deploy and configure Spack at the target systems, we use Ansible. Both Ansible and Spack feature deep integration of YAML and JSON formats along with a high extensibility via a wide range of built-in and external modules. It makes these tools a perfect combination not only for automated fully-reproducible software installation, but also for documenting hardware and software configurations along with the installation process uniformly in a human readable format, easy to post-process. Last, but not least, automated deployment approach with Ansible and Spack depends on the software which is naively installed on the most of existing HPC and HPDA platforms, and requires Internet access only on the workstation with the Ansible manager (see Table 9).

Dependencies	Local workstation	Target HPC/HPDA system
Python 2.6+ or 3.5+	**AS**	**S**
extra modules of Python	**A**	
bash		**AS**
ssh	**A**	
a C/C++ compiler for building		**S**
make		**S**
tar, gzip, unzip, bzip2, xz	**A**	**S**
patch		**S**
git and curl for fetching along with internet access	**S**	

Table 9. Prerequisites for launching HiDALGO automated deployment approach with Ansible and Spack: "A" corresponds to dependency imposed by Ansible, "S" corresponds to dependency imposed by Spack.

Source codes of the HiDALGO AD/SCM scripts are uploaded to the GitLab repository: https://gitlab.com/eu_hidalgo/hidalgo_integration. This repository consists of two folders:

- `ansible` which contains Ansible scripts for deploying and configuring Spack on the HiDALGO HPC/HPDA infrastructure, and
- `spack-configs` which contains configuration files of Spack.

Both folders are supplemented with the detailed `README.org` files.

Content of the Spack configs folder follows guidelines of the *Extreme-scale Scientific Software Stack* (E4S) project. Namely, it includes three subfolders: subfolder `repos/hidalgo` with the custom Spack packaging scripts, subfolder `site-scopes` with the configurations for the target HPC/HPDA systems, and subfolder `stacks/sna` with the use case software stacks.

Subfolder `repos/hidalgo` serves to store Spack packages for HiDALGO which are either configured unconventionally or are not the part of the built-in Spack package repository yet. In particular, it contains definition of package `libosrm`, an OSRM backend library which is used for routing in location graph extraction codes of the migration pilot.

Subfolder `site-scopes` contains configurations for the clusters Hawk, Vulcan, and Eagle/Altair, which constitute the core HPC and HPDA hardware infrastructure of the HiDALGO. Configurations files for each cluster are stored in a separate subfolder which includes:

- `config.yaml` with generic configuration options,
- `compilers.yaml` with compiler definitions,
- `packages.yaml` with build customization, and
- `upstreams.yaml` with installation trees of external `Spack` instances (if applicable).

These files are prepared manually and regularly updated within HiDALGO in case of changes at the default configurations and software stacks of the HiDALGO clusters. Config files `mirrors.yaml` and `repos.yaml` are configured automatically by Ansible scripts as discussed below. After configuring, `repos.yaml` stores link to the copy of the HiDALGO repository `repos/hidalgo` at the target cluster, while `mirrors.yaml` holds link to the local automatically created off-line mirror for the clusters with Internet ban (Hawk and Vulcan) and link to the default on-line Spack mirror for the clusters with open Internet access (Eagle).

Subfolder `stacks` contains definitions of software environments for the use cases.

We use the Python3 Flee code to perform migration simulations. Since Flee is not originally supported by Spack, we developed a simple Spack package for Flee. For forecasting purposes we also need to generate a conflict scenario, this can be done using the Python3-based Flare code.

For modelling COVID-19 spread, we use the FACS code. The FACS code relies in a wide range of input files, which are described in detail in https://facs.readthedocs.io.

The simulation of the SN use case is written in Python3; as is the code of the validation program (KPM eigenvalue tool). The code integrates some Python packages, mainly mpi4py, scipy, numpy and pandas.

The UAP use case can be viewed as a multi-layer simulation tool containing a wide variety of programming environments as, for example, Python, Java or Bash, with different open source and proprietary software as well. It is thus recommended to use singularity container environment to run the UAP. A simple shell script using Spack was developed to install all necessary software tools to run the UAP system wide as well.

Ansible inventories, scripts, and roles are collected in the folder `ansible`. HiDALGO clusters and their settings are defined in the file `01-clusters.yaml` of the subfolder `inventory`.

Folder `ansible` contains two core Ansible scripts `show_hardware.yml` and `sites.yml`. Script `show_hardware.yml` gathers basic facts about the clusters such as hardware configurations and OS setup. This is done by calling module `ansible.builtin.setup`.

Script `sites.yml` deploys and configures Spack on the HiDALGO clusters. At first, it fetches Spack sources from the official GitHub repository and then deploys and configures it with the files from the folder `spack-configs` (discussed above). In order to install and configure Spack package manager, we developed Ansible role `spack`. This role installs Spack in the local directory on the HPC clusters taking into account whether the users have internet access from the cluster or not. On clusters without internet access, it automatically creates local off-line mirrors with software packages required by the pilots. The role supports two models of deployment with local off-line mirrors: deployment with a common mirror for all clusters (see Figure 12) and deployment with mirrors specific for each cluster (see Figure 13).

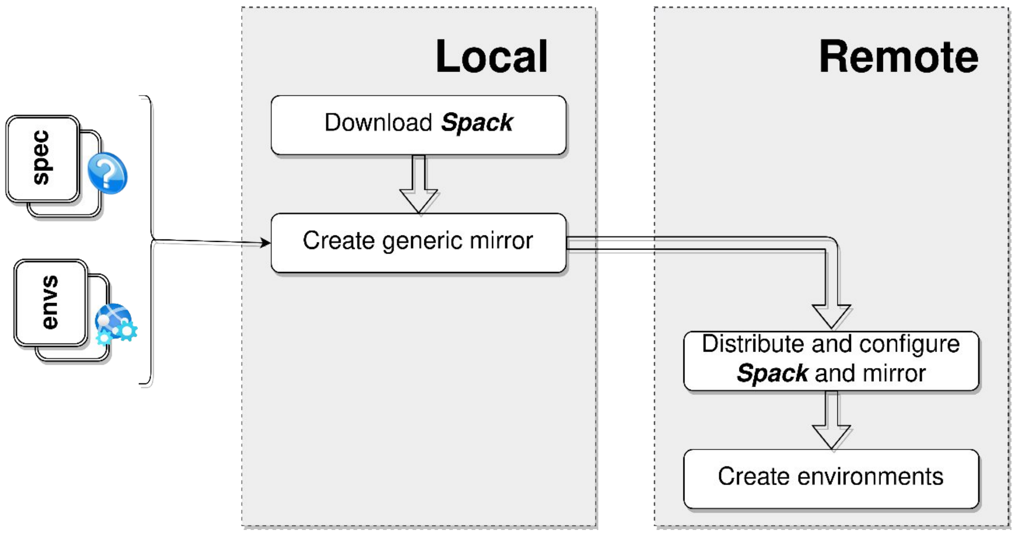


Figure 12: Spack deployment workflow with a common mirror.

In the first approach illustrated in Figure 12, we use pilot software environments to create a single mirror with all possible dependent packages with the predefined depth of mirrored versions at the local workstation. Afterwards, we copy this mirror to the HPC clusters, as well as install and configure Spack in there. The drawback of this approach is that the mirror can be large in side and include many packages which are already installed at the remote clusters.

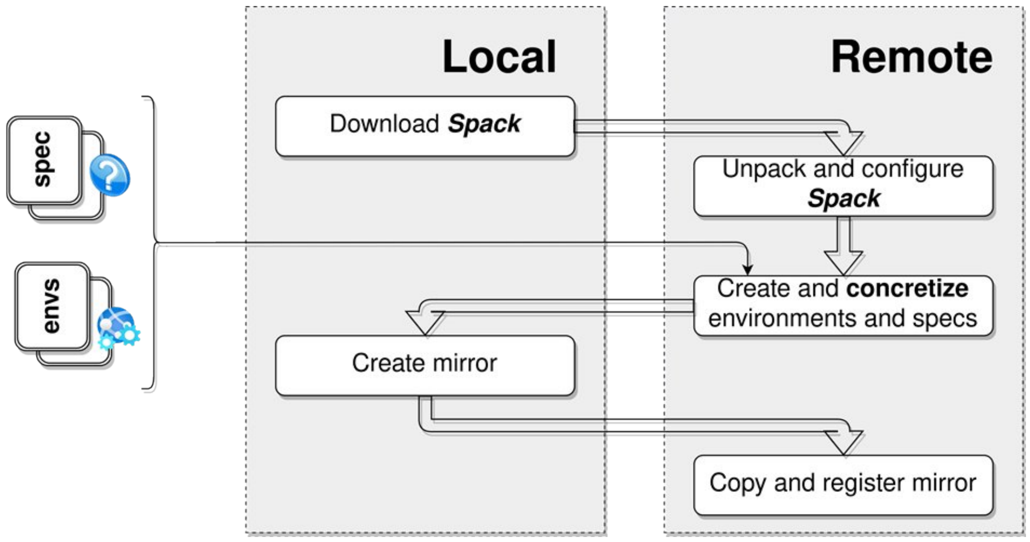

Local
Remote
spec
envs
Download Spack
Unpack and configure Spack
Create and concretize environments and specs
Create mirror
Copy and register mirror

Figure 13: Spack deployment workflow with the mirrors optimal for each remote system.

In the second approach as illustrated in Figure 13, we start by installing and configuring Spack with the pilot software environments on the remote HPC clusters. Afterwards, we copy `lock`-files of the concertized environments to the local workstation with Internet access and fetch the mirrors separately for each cluster taking into account information about pre-installed software from the `lock`-files. In the final step, these mirrors are copied to and registered on the corresponding HPC clusters.

4.2.1.2 CI/CD and Testing

HiDALGO uses GitLab-CI as a tool for implementing the CI/CD mechanism for the use cases software. In order to make it work, we created a GitLab repository https://gitlab.com/eu_hidalgo/use_cases, which contains the definition of the GitLab-CI pipeline for the pilots (`.gitlab-ci.yml`) along with the references to the HiDALGO software for testing. HiDALGO packages are referenced as Git submodules. When executing the GitLab-CI pipeline, GitLab fetches the most recent versions of the HiDALGO packages and runs tests with them. It is implemented by setting up variable `GIT_SUBMODULE_STRATEGY` to `recursive` and calling for the recursive update of the Git submodules in the section `before_script` of `.gitlab-ci.yml`:

```
variables:
  GIT_SUBMODULE_STRATEGY: recursive

before_script:
  - apt update && apt install git tree -y && git version
  - git submodule update --remote --recursive
```

For Flee (migration) we provide a range of functional, and integration tests. The most important test suite is the auto validator, which is a FabFlee-based tool that relies on a Verification and Validation Pattern from the VECMA project [40]. This script automatically runs a Flee simulation for each validation scenario (there are at least 7 currently in existence) and calculates the averaged relative difference between the Flee results and the validation data from UNHCR. In addition, there is a wide range of test scripts available, to test functionalities ranging from border closures to multiscale coupling. These tests can be found in the tests/ subdirectory (https://github.com/djgroen/flee/tree/master/tests) and in (https://gitlab.com/eu_hidalgo/use_cases/-/tree/master/migration).

For FACS we use a similar approach, although the number of validation scenarios in the auto validator is much smaller (2 at time of writing), due to the young age of the code. The integration testing infrastructure for FACS is much simpler, as the code is still subject to frequent change (excessive test infrastructure can reduce the speed of adaptations in earlier stages of code development). However, it is present and can be found in the test/ directory as well (https://github.com/djgroen/facs/blob/master/tests/test_facs.py).

For the Social Network simulation, we have both a small number of unit tests as well as a simple functional smoke test serving as an integration test. As in the case of FACS, the limited number of the tests is due to the high volatility of the simulator code and the young age. Indeed, the main purpose of the testing framework is to aid in development, so all the tests are run automatically on different Python and MPI variants using tox-conda (https://github.com/tox-dev/tox-conda) after every modification on GitHub.

For the Urban Air Pollution simulation, in particular for the traffic emission calculation workflow, some complex calculations were implemented. One of these heavy calculations is executed in the traffic-emission-calculator tool, which implements a Copert 5 hot emission model with high spatial and time resolution. The corresponding java implementation has been almost completely covered by unit tests.

The matcher module finds intersections between traffic segments CFD mesh elements. The critical parts of the algorithm have been covered with unit tests. The Cdf-emi-generator is another computation-critical tool, which generates CFD mesh connected emission files based on matcher results and traffic emissions. For this tool unit test has been implemented to check the correct calculation.

Almost continuous manual integration testing is conducted for the whole workflow, for different area-of-interests, traffic flows and configurations. The workflow is built up from microservices communicating with files.

The implemented unit tests and the continuous manual monitoring of the internal files guarantee high quality results of the workflow, and prevents any regression in case of the development process.

4.2.2 Portal Components

The final version of the web portal extends the functionalities of the predecessor to run use-cases applications with the graphical web submission forms and integrates all the services with Single-Sign-On to provide coherent access through its frontend. Portal components and their configurations are changed from the predecessor, which is detailed in the rest of the chapter to highlight the new features in each component for satisfying the requirements of the portal developments. For more details on these components, we refer to D5.7 [10].

(a) Askbot

Askbot is an open-source software used to create question and answer oriented user forums. It provides the possibility to exchange information publicly and to allow the community and helpdesk members to help with common issues. Access to the questions and knowledge is public, however participating in the discussions requires an account. Thanks to Keycloak SSO integration any HiDALGO portal user can authenticate and start asking and answering questions. This service allows the creation of a permanent, open knowledge base.

(b) CKAN

The purpose of CKAN is to register datasets, facilitate the search of datasets, and finally provide access to these datasets. Besides providing these core functionalities related with data storing, CKAN also allows grouping of datasets, creating organizations, metadata management, and relationship management of data, it can handle different data formats, can harvest external data sources and provides a shared pool of data where all can benefit from the publicly available collection of data of other users which covers most if not all of the pilots' requirements. CKAN is integrated with the HiDALGO SSO Keycloak instance. The data can be uploaded to the CKAN using web interface, REST API and dedicated scripts (also using transfer protocols like GridFTP and SCP).

(c) Cloudify

Cloudify is a default workflow manager for defining pilots' workflow and it is the backbone for executing the pilots' application in the HiDALGO portal. Cloudify changed the versioning name from the date of release to the semantic versioning and the latest version is 5.2.1 at the time of writing this report. The latest version of Cloudify is installed in the production VM to support the latest Croupier plugin with Python 3 programming language. the HLRS infrastructure is not directly accessible from the HLRS cloud due to the network policy so that the production VM of Cloudify is located in the PSNC cloud for providing an interconnection with the HLRS Hawk and Vulcan infrastructures. All of the HiDALGO infrastructures are successfully federated with Cloudify for executing the pilot applications from pre-processing to post-processing by defining their workflow in the Cloudify-TOSCA blueprint. Different HPC centre has its user management in their infrastructure, so it is federated by providing single HiDALGO's user management practices by using Keycloak and Vault software. Keycloak portal login and Vault secret value are used for mapping the HiDALGO's portal users to different infrastructures automatically and securely to provide a complete federation of the infrastructure. The workflow of all the pilots is defined in the Cloudify-TOSCA blueprint and it is successfully integrated with the portals to submit jobs easily through the GUI web forms.

Vault is used for storing infrastructure credentials, due to the limitation of Cloudify single user (admin) authentication. ECMWF Weather Cloud is integrated with Cloudify for pre-processing the weather data.

(d) Matchmaking

Matchmaking is a REST service for referring users based on their interests. The previous version of Matchmaking was based on the Geometric Mean Algorithm, which is extended in the latest version to use the Machine learning technologies. RDFGraph is generated from the user's information of profession, location and their initial answers, which is used for referring the users accurately. The new version of Matchmaking extends its core functionality by using the latest Machine Learning technologies and the user's information. Matchmaking was earlier based on the Django default authentication, which is extended to support SSO authorization by using **mozilla-django-oidc**. The REST end-points are currently secured with the Keycloack OpenIDC (Open ID Connect) protocol and it is the entry point for portal integration. The portal has successfully integrated with the Matchmaking service by consuming the API in the portal backend through OpenIDC protocol and has provided an easy GUI for interactive web access at the portal frontend.

(e) Moodle

Moodle is a platform for offering training and its material in a single place to the end-users. HiDALGO favicon image support, SSO login, disable guest login, enable analytics module and enable the data retention policy are the few changes introduced in the configuration of Moodle application to provide an authentic training service from the HiDALGO portal. The analytics module is helpful for the course instructors to get the prediction of the possible students drops out from each course, which is useful for retaining the student by adapting the course content based on the student's overall needs. The HiDALGO portal integrated the service successfully through the SSO login by using the OAuth2 protocol. New HiDALGO courses are created by the course instructors for disseminating the HiDALGO technologies from the first version of the portal to use the services successfully.

(f) Jupyter Hub

JupyterHub is a web-based interactive development environment for developers to analyse the outcomes of their simulations. The tool is seamlessly interconnected with the HiDALGO CKAN data management system to provide access to stored results and allow their analysis using custom scripts and their visualization using python libraries for gaining more insights from the outcome. The service is mainly provided for programmers to perform analysis using simulation output data, whereas non-programmers can use our default visualization tools (COVISE or Visualizer) to visualize the simulation outcome interactively in the portal.

The initial installation of JupyterHub has been extended and now supports R and C/C++ in addition to Python, in order to attract a wide range of programmers. Leveraging its support for OAuth2 authorization, JupyterHub has also been integrated with the HiDALGO Single-Sign-On (SSO) infrastructure, allowing users to login through the portal using the same credentials as for any other service provided by the HiDALGO. More details can be found in deliverables D5.6 [9] and D5.7 [10].

(g) Visualizer

Visualizer is a web-based visualization tool for tabular data using coordinated multiple views. It enables users to easily create new dashboards by selecting interesting data fields and suitable visualizations. Besides visualizing data, users can apply analytical workflows including ML algorithms. So far, clustering and outlier detection are integrated in Visualizer. Visualizer can be easily integrated to any other website, including the portal. Also, user-configured dashboards can be deployed, as all dashboard information is stored in the URL. Detailed descriptions about Visualizer's functionality have been provided in D3.3 [1], D3.4 [4], and D3.5 [5]. Details about how to integrate Visualizer and dashboards in other web applications have been discussed in D5.6 [9] and D5.7 [10].

(h) COVISE

COVISE (COllaborative VIsualization and Simulation Environment) is an open-source visualization software environment developed at HLRS. It enables the integration of simulations, post-processing of data and their visualization and facilitates the combination of heterogenous data. COVISE is modular in structure as each processing step is implemented as a separate processing module. These modules can then be distributed to different heterogeneous machine platforms to improve the performance when processing HPC data. Moreover, COVISE is designed for collaborative working, allowing multiple users to work jointly via network infrastructure. Rendering in COVISE is enabled through a dedicated module. This module, OpenCOVER, supports rendering in various virtual environments, such as CAVEs, head-mounted displays or powerwalls. OpenCOVER enables the examination of data in an immersive, interactive manner. Besides, also Augmented Reality is supported.

(i) Keycloak

Keycloak is the solution adopted in order to provide a Single-Sign-On service. This tool is integrated with all those tools that allow the delegation of user access and authorization by means of standards like OpenId. We have created a specific realm for HiDALGO, in which all the Portal components are registered as clients, so the Keycloak instance is able to provide service to all of them.

When a user wants to login from the Frontend, it is forwarded to Keycloak, that will provide the security token that can be used during the current session. Thanks to such token, it is possible to access other tools such as the Moodle, the Jupyter Hub, the Askbot, the Wiki and other Portal services managed by the backend.

It has been configured in such a way that users can self-register to the HiDALGO Portal, including the captcha feature to avoid bots.

(j) Portal Frontend and Backend

The Portal backend is a Python-based service that interacts directly with the Cloudify Orchestrator. It takes care of collecting input information from the Frontend and sending the corresponding calls to Cloudify through the REST APIs. It has the capability to authenticate users with Keycloak before doing any call to Cloudify and it is able to manage users, applications, instances, executions and infrastructures.

On the other hand, the Frontend is a component implemented with Angular that provides the common GUI of the Portal. It integrates all the functionalities (in most of the cases, through iFrames): from the data catalogue to the training tools, including the access to the visualization tools, the ticketing system and the Jupyter hub. It is integrated with all the components, relying the user management and Single-Sign-On to the Keycloak. Moreover, it implements the GUI to enable the management and execution of applications, providing easy to use interfaces with the Orchestrator (through the Django Backend service). There is an ongoing development to integrate it with Vault as well, in such a way that the storage of users' credentials will be much easier.

The Frontend has been designed in such a way that it is easy to use, and it provides a one-stop-shop for HiDALGO.

(k) Marketplace

As a way to facilitate the commercialization of services in HiDALGO, we are integrating a marketplace in the Portal. Such marketplace is a solution for e-commerce that allows publishing services (mainly, applications and data) that can be offered for free or for a certain price. Such platform provides the features required for billing and payment, associated to the products published in the marketplace. We considered OpenCart as the solution to select, but after the latest updates, and since it seems to be issues in the integration with the SSO approach, we consider using WooCommerce, a WordPress-based platform that can provide the required features.

(l) Wiki.js

Wiki.js is a community editing platform for sharing information within the consortium, which is extended to offer the service for the external end-users by sharing the public wiki pages. The application is successfully integrated by using SSO authentication with the Keycloak OpenIDC login and enabled the public wiki page for end-user access. The wiki pages are accessed based on the role assigned to the login user, which is default assigned to access the public wiki pages from the HiDALGO portal by using the Keycloak authentication. The system admin is allowed to change the roles manually for the consortium members and provide an access to the private wiki pages. The public wiki helps to foster the external community building and allows the end-users to share the information on a public platform to establish sustainability within the project.

(m) Zammad

Zammad is a support ticket system for addressing customer queries securely by adapting the process of ITSM service management. This is installed for both internal and external customer access to resolve issues on time with high quality by the HiDALGO support team. Zammad is installed to support the following features for providing the services:

- Users can raise the support ticket using the official E-mail IDsupport@hidalgo-project.eu and the web tool (https://support.hidalgo-project.eu/)
- Automatic acknowledgement emails are sentno-reply@hidalgo-project.eu
- Tickets are classified using multiple tags (internal, external customer help, bug, etc.) to address different tickets based on the SLAs (Service Level Agreements)
- Two-level support is given to the customer, so Basic (L1) and Expert (L2) support is provided to increase customer satisfaction by resolving tickets on time
- Automatic ticket escalation is enabled to send the escalation E-mail to the respective support agents, and escalation manager to manage quality and SLAs

4.2.2.1 Deployment and Access

The portal is an entry point for accessing technologies and services developed in the different work packages, so all the components are required to integrate with the portal by using the common integration interface between the portal and the component.

WP3 offers the CKAN data management system and Visualization tools, both are the standalone web applications to integrate with the portal by using the SSO (Single Sign-On). OAuth2 or OpenID is the standard protocol used for enabling SSO, so those services are required to enable the authorization with those protocols. If there is a complication to enable SSO authentication for a specific service, then the iFrame is considered as an alternative to integrate with the portal. CKAN is successfully enabled OpenID protocol to establish SSO authentication with the portal.

Portal integrated Cloudify workflow manager, which is considered as an integration interface between the **use-case applications** and the portal. Cloudify offers the REST API service, which is considered as the integration interface between Cloudify and the portal by following the Service Oriented Architecture (SOA). Cloudify does not provide SSO login, due to the limitation of the community version to support single admin users, so it is integrated hardcoded with portal backend to login with the admin credentials. Cloudify blueprints are considered as an integration interface between the pilot application and portal backend. We integrated all the use-case applications with the portal by using the Cloudify workflow manager and submit jobs by using the interactive web forms through the portal. The user's critical information of private key is not allowed to store in the Cloudify with the blueprint and Key-value management, so it is stored in the Vault and access it through Keycloak SSO login. Vault is a secret key value management system, which is integrated with the Keycloak login and Cloudify admin login to provide a secure login mechanism for executing the pilots' application, which is the integration interface with the portal backend for Cloudify and substitute the incapability of supporting the SSO login in Cloudify.

Besides CKAN, Visualization, and Cloudify, WP5 offers additional services such as the Askbot, Zammad, Jupyter Hub, Wiki, Marketplace, Matchmaking and Moodle. All those services are standalone web applications, which are integrated with the portal by using SSO authentication. Matchmaking is a social networking application to connect similarly interested people, and it is running as an individual REST service. Portal integrated the Matchmaking service by using the REST interface and display the REST data in a graphical interface for easy access by users.

Portal integration infrastructure is provided with Jenkins for the components- and system-level integration. Jenkins automates the software deployment and testing to help the integration process. Component-level integration interface is defined and documented, which is used as a reference for integrating multiple components one by one in the integration infrastructure. We defined the Jenkins pipeline for each component and the whole system in order to automate the integration and testing of the HiDALGO infrastructure.

4.2.2.2 Authentication and Authorization

The authentication and authorization feature in HiDALGO is managed by the Keycloak component. There are two instances of the Keycloak, one for integration and another one for production.

In both cases, the realm 'HiDALGO' has been created and configured specifically for the project. Such configuration includes the following features:

- We have added as clients all the HiDALGO components of the Portal that allow for the integration with the Single-Sign-On feature: Askbot, Frontend, Backend, Moodle, Wiki, Jupyter notebooks, Visualizer, CKAN, Matchmaking, ECMWF WCDA, Zammad.
- We enabled the self-registration for users, with e-mail validation (now Keycloak can send the validation emails).
- We modified the registration workflow, so now it includes Google Captcha v2 as a way to avoid bots registering.
- We have created several groups for user management: Developers and Guests. Depending on the role of the users, they will belong to a group or another (or none), so we can filter the access scope at the tool side.

All the tools registered as clients have included a specific configuration in order to enable the Single-Sign-On capabilities, sometimes requiring the installation and integration of specific libraries or extensions.

Additionally, we are setting up a Vault instance that will be used for managing users' credentials (those keys that must be used to access HPC/Cloud infrastructures and other protected APIs, like the CKAN REST API). Vault will only store credentials (permanent or temporarily, depending on the users' preference), and Keycloak will manage the access to Vault through a temporary security token that can be used only once. This solution is described in full detail in D3.5 [5].

4.2.2.3 CI/CD and Testing

Component development, integration and testing are interlinked with each other to follow the iterative process, so different stages of testing are conducted to ensure the quality of the components at the different stages. Unit, integration, system, alpha and beta tests are conducted to test the components from the perspective of functional, integration and user experience. The component-level test case scenarios are defined for the Cloudify croupier plugin, which is automated with tox and python-unittest framework to perform the unit tests. One or two components are integrated and the test is conducted at the level of their interconnection, which is performed for the portal backend and other services at the level of SSO login, portal backend and other pilots at the level of Cloudify blueprint. All the components are integrated with the portal frontend to provide a complete system, which is tested at the system level by using black-box manual tests to ensure the correctness of functionality implementation. The beta tests are conducted for the portal to test the system from the user perspective and improved the usability of the portal based on the test outcomes. The product artefacts (source code, binary executable) and documentation are shared with the beta testers to ensure the usability of the individual components from the user perspective. ReachOut! [39] is a European project carrying out beta testing, and we had a strong collaboration with them to perform the beta tests with the real user environment to check both functionalities and user experience.

5 Conclusions

This deliverable serves as final report for work package 6 named "Requirements Evolution and Component Integration" covering three main aspects of the project: (i) Requirements Capture & Evaluation, (ii) AI-enabled use case workflows and (iii) the data & component integration. Deliverable D6.6 focuses on reporting results from year 3 of the project, and also concludes the entire work package.

On the one hand, this relates to the role AI has played over the past three years. Exploring ways to adapt existing methods to concrete challenges and problem settings brought the communities closer together, thereby broadening our views and allowing the creation of new ideas and possibilities. Our efforts resulted in seven AI components to support the use case workflows - documented by several joint publications with our partner institutions.

On the other hand, WP6 has orchestrated the integration of static and streaming data sources as well as the components; WP6 has taken care of integrating the developments of WP3, WP5 and WP6 itself as well as the use case scenarios (WP4) into an entire solution - into a Centre of Excellence accessible via our HiDALGO portal.

This conclusion (and parts of D6.6) should also serve as a *lesson learnt* over the past three years to also document approaches and decisions.

First, the gathering and compiling of necessary requirements (hard/soft) at the beginning was cumbersome and required a real effort from all partners. A procedure to recommend is to align requirements, objectives, KPIs of the project, tasks and responsible partners as early as possible in the project. Underrepresented tasks or objectives can thus be identified and corrected at an early stage of the project. Equally important as the initial phase of requirements capture, is the constant monitoring of established requirements and their deadlines. We therefore recommend installing a dedicated requirements monitoring process to take care of constantly reminding partners.

Second, we noticed that possible ways in which AI can support the use cases requires getting familiar not only with the requirements of each use case, but also with the inner workings of the current implementations. This might have required more time than was initially planned; the AI support had to be outlined in D6.3 [14], which was due in M6. Our results were most successful, if the AI support was closely concerted with the use case owners and was developed in multiple iterations with them. This shows that intensive bilateral meetings, starting already early in the project duration, are essential for success.

Third, in course of our data and component integration efforts, we came to appreciate a combination of Ansible and Spack allowing the deployment of HPC-compliant software in a simple and elegant way on a wide variety of HPC/HPDA clusters and testbeds with different usage policies and security restriction; both tools can be configured with YAML scripts. Moreover, they have modular design conveniently extensible via Python APIs. These characteristics make Ansible and Spack convenient to couple and to maintain. To build elaborate CI/CD-pipelines, GitLab-CI has turned out to be a good, simple and free of charge instrument, especially for open-source projects for they need not to be hosted in GitLab. On the negative side, GitLab-CI is less comfortable than Jenkins for solving more intricate CI/CD tasks which involve one's own infrastructure. Providing both Jenkins and GitLab-CI might be a good compromise to mitigate Jenkins' complexity and GitLab-CI's lower flexibility.

Annex A

Title: “Markov Aggregation for ABM-Based Refugee Simulations”

Author: “Bernhard C. Geiger”

For details on Markov aggregation for the Migration use case, we refer the reader to Annex File “Annex_A_MarkovAggregation.pdf”.

Note that this preprint is not (yet) public, hence will not appear on the HiDALGO website.

Annex B

Title: "Generating simple directed social network graphs for information spreading"

Abstract: "Online social networks have become a dominant medium in everyday life to stay in contact with friends and to share information. In Twitter, users can establish connections with other users by following them, who in turn can follow back. In recent years, researchers have studied several properties of social networks and designed random graph models to describe them. Many of these approaches either focus on the generation of undirected graphs or on the creation of directed graphs without modelling the dependencies between directed and reciprocal edges (two directed edges of opposite direction between two nodes). We propose a new approach to create directed social network graphs, focusing on the creation of reciprocal and directed edges separately, but considering correlations between them. To achieve high clustering coefficients (which is common in real-world networks), we apply an edge rewiring procedure that preserves the node degrees. Our model is based on crawled directed graphs in Twitter, on which information with respect to a topic is exchanged or disseminated. Similar to other real-world networks in different areas, these graphs exhibit a high clustering coefficient and small average distance between a random pair of nodes. However, their degree sequences do not follow a power law, but are rather $\chi 2$-distributed. We analyse and compare the structure of the crawled and the created graphs, and simulate certain algorithms for information dissemination and epidemic spreading. The results show that the created graphs exhibit very similar topological and algorithmic properties as the crawled real-world graphs, providing evidence that they can be used as surrogates in social network analysis.

Furthermore, our graph model is highly scalable, which enables us to create graphs of arbitrary size with (almost) the same properties as the corresponding real-world networks."

For details on synthetic graph generation for the Social Networks use case, we refer the reader to Annex File "Annex_B_GraphGeneration.pdf".

Note that this publication is currently undergoing a double-blind review process – author names are thus omitted.

Annex C

Title: “Minimum Spanning Trees and the Inference of Message Cascades”

Author: “Bernhard C. Geiger”

For details on message cascade inference for the Social Networks use case, we refer the reader to Annex File “Annex_C_MessageCascades.pdf”.

Note that this preprint is not (yet) public, hence will not appear on the HiDALGO website.

References

[1] HiDALGO, D1.4 - Final Project Report. 2022.

[2] HiDALGO, D2.4 - Final Report on Component Exploitation and Sustainability Strategy. 2022.

[3] HiDALGO, D3.3 - Intermediate Report on Implementation and Optimisation Strategies. 2021.

[4] HiDALGO, D3.4 - Intermediate Report on Benchmarking, Implementation, Optimisation Strategies and Coupling Technologies. 2021.

[5] HiDALGO, D3.5 - Final Report on Benchmarking, Implementation, Optimisation Strategies and Coupling Technologies. 2021.

[6] HiDALGO, D4.2 - Implementation Report of the Pilot Application. 2021.

[7] HiDALGO, D4.3 - Implementation Report of the Pilot Applications Year 2. 2021.

[8] HiDALGO, D4.4 - Final Implementation Report of the Pilot and Future Applications. 2022.

[9] HiDALGO, D5.6 - Second HiDALGO Portal Release and System Operation Report. 2021.

[10] HiDALGO, D5.7 - Final Portal Release and System Operation Report. 2021.

[11] HiDALGO, D5.8 - Final Benchmark Results for Innovative Architectures. 2021.

[12] HiDALGO, D6.1 - Requirements Process and Results Definition. 2019.

[13] HiDALGO, D6.2 - Workflow and Services Definition. 2019.

[14] HiDALGO, D6.3 - Artificial Intelligence Approach. 2019.

[15] HiDALGO, D6.4 - Initial Report on Requirements, Components and Workflow Integration. 2019.

[16] HiDALGO, D6.5 - Intermediate Report on Requirements, Components and Workflow Integration. 2020.

[17] HiDALGO, D7.5 - Final Report on Community Building, Event Management, Collaboration and Training. 2022.

[18] HiDALGO, Grant Agreement, 2018.

[19] Bansal, S., & et al. (2009), Exploring biological network structure with clustered random networks, *BMC Bioinformatics 10*(405).

[20] Chaturantabut, S. & Sorensen, D. (2010), Nonlinear Model Reduction via Discrete Empirical Interpolation, SIAM Journal on Scientific Computing, 32(5), 2737–276.

[21] Chung, F., & Lu, L. (2002), Connected Components in Random Graphs with Given Expected Degree Sequences, *Annals of Combinatorics 6*(2), 125-145.

[22] Geiger, B. C., & Al-Bashabsheh, A. (2020), On Functions of Markov Random Fields, *In 2020 IEEE Information Theory Workshop*, 1-5.

[23] Gamblin, T., LeGendre, M., Collette, M., Lee, G., Moody, A., de Supinski, B., & Futral, W. (2015), The Spack Package Manager: Bringing Order to HPC Software Chaos, *In Supercomputing (SC'15)*, 15-20.

[24] Molloy, M., & Reed, B. (1995), A critical point for random graphs with a given degree sequence. *In Random Structures and Algorithms 6*(2-3), 161-180.

[25] Rosvall, M., & Bergstorm, C. T. (2008), Maps of random walks on complex networks reveal community structure. *In PNAS*, *105*(4), 1118-1123.

[26] Schweimer, C., & et al. (2021), A route pruning algorithm for an automated geographic location graph construction. *In Scientific Reports*, *11*(1).

[27] Taylor, S.J. & Letham, B. (2017), Forecasting at Scale, *PeerJ Preprints*.

[28] Toth, C., Helic, D., & Geiger, B. C. (2020), SynWalk – Detecting Communities with Synthetic Random Walk Models, *in preparation*.

[29] Velimsky, J., & et al. (2020), Message spread on Social Media. Comparing the FPOE and NEOS during the election campaign in the 2019 Austrian National Council Elections, *Tag der Politikwissenschaft 2020.*

[30] Volkwein, S. (2013), Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling. Lecture notes, University of Konstanz, 2013.

[31] AutoARIMA, https://alkaline-ml.com/pmdarima/, retrieved 2021-11-30.

[32] OpenStreetMap, http://www.openstreetmap.org, retrieved 2021-11-30.

[33] Open Source Routing Machine, https://github.com/Project-OSRM/osrm-backend, retrieved 2021-11-30.

[34] OSMnx, https://osmnx.readthedocs.io/, retrieved 2021-11-30.

[35] ExtractMap, https://github.com/djgroen/ExtractMap, retrieved 2021-11-30.

[36] Synwalk, https://github.com/synwalk, retrieved 2021-11-30.

[37] Pokec Social Network Graph, https://snap.stanford.edu/data/soc-pokec.html, retrieved 2021-11-30.

[38] Spack Documentation, https://spack.readthedocs.io/en/latest/, retrieved 2021-11-30.

[39] ReachOut! Project, https://www.reachout-project.eu/, retrieved 2021-11-30.

[40] VECMA Project, https://www.vecma.eu/, retrieved 2021-11-30.