

D3.5 – DOCUMENTATION AND SOFTWARE ON NEW METHODS, TOOLS AND MECHANISMS FOR RELEASE 2 OF THE PORTAL

Grant Agreement	676547
Project Acronym	CoeGSS
Project Title	Centre of Excellence for Global Systems Science
Topic	EINFRA-5-2015
Project website	http://www.coegss-project.eu
Start Date of project	October 1, 2015
Duration	36 months
Deliverable due date	31.03.2017
Actual date of submission	29.03.2017
Dissemination level	Public
Nature	Report
Version	1.5.0
Work Package	WP3
Lead beneficiary	HLRS
Responsible scientist/administrator	Sergiy Gogolenko, Michael Gienger, Ralf Schneider
Contributor(s)	Sergiy Gogolenko (editor), Michael Gienger (editor), Burak Karaboga, Francisco Javier Nieto De Santos, Michał Pałka, Oskar Allerbo, Fabio Saracco, Andrea Rivetti, Marcin Lawenda, Wolfgang Schotte

Internal reviewers	Sarah Wolf, Tiziano Squartini
Keywords	portal, global system sciences, HPC, centre of excellence, CoeGSS, data management, synthetic population, synthetic network, data analysis, ABMS, CKAN, Moodle, AskBot, LDAP
Total number of pages:	36

Copyright (c) 2016 Members of the CoeGSS Project.



The CoeGSS (“Centre of Excellence for Global Systems Science”) project is funded by the European Union. For more information on the project please see the website [http:// http://coegss-project.eu/](http://coegss-project.eu/)

The information contained in this document represents the views of the CoeGSS as of the date they are published. The CoeGSS does not guarantee that any information contained herein is error-free, or up to date.

THE CoeGSS MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, BY PUBLISHING THIS DOCUMENT.

Version History

	Name	Partner	Date
From	Sergiy Gogolenko Michael Gienger Ralf Schneider	USTUTT- HLRS	March 9, 2017
Version 1.0	for 1 st internal review	USTUTT- HLRS	March 10, 2017
Reviewed by	Sarah Wolf	GCF	March 17, 2017
	Tiziano Squartini	IMT	March 13, 2017
Version 1.2	for 2 nd internal review	USTUTT- HLRS	March 22, 2017
Reviewed by	Sarah Wolf	GCF	March 24, 2017
	Tiziano Squartini	IMT	March 23, 2017
Version 1.5	for submission	USTUTT- HLRS	March 27, 2017
Approved by	Coordinator	UP	March 28, 2017

Abstract

The portal is a single entry point for all CoeGSS stakeholders. It provides a broad variety of services to GSS and HPC communities. The portal is one of the major contributions of the CoeGSS project.

This document addresses achievements in implementing services specified in the deliverable D3.2 as far as they are realized to be integrated in release 2 of the portal, planned at month 20, the overall Centre's workflow.

In this text, we not only describe fully-operating portal components and trial interfaces to the software components included into release 2 of the portal, but also discuss ideas and plans regarding integration of the software components which are still under development. In contrast to D3.2, all over the document we emphasize the implementation of interfaces between the tools and the portal, not on the implementation of the tools themselves. In addition, we discuss potential use cases.

Table of Contents

- Glossary 5
- 1 Introduction..... 7
- 2 Current state of the portal 8
- 3 Documentation of operating components 10
- 4 Software integration 18
- 5 Software outlook..... 29
- 6 Summary 34
- References..... 35
- List of tables 36
- List of figures 36

Glossary

3D	Three-dimensional space.
ABM	Agent-Based Model
ABMS	Agent-Based Modeling and Simulation
Apache Spark	an open-source cluster-computing framework
API	Application Programming Interface
AskBot	a popular open source Q&A Internet forum
CAVE	Cave Automatic Virtual Environment (an immersive virtual reality environment)
CKAN	Comprehensive Knowledge Archive Network
CoE	Centre of Excellence
CoeGSS	Centre of Excellence for Global Systems Science
COVISE	Collaborative Visualization and Simulation Environment
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CSV	Comma-Separated Values file format
Cuda	a parallel computing platform and API model created by Nvidia
D	Deliverable
DCAT	Data Catalog Vocabulary
DDR (DDR SDRAM)	Double Data Rate Synchronous Dynamic Random-Access Memory
Django	a free and open-source web framework, written in Python
DSL	Domain-Specific Language
EC	European Commission
Eclipse CDT	Eclipse C/C++ Development Tooling
GB	Gigabyte
GB/s	Gigabytes per second
GDDR	Graphics DDR SDRAM
GHz	Gigahertz
GIS	Geographic Information System
GitHub	a web-based VCSs repository and Internet hosting service
GPU	Graphics Processing Unit
GSS	Global Systems Science
HDF5	Hierarchical Data Format, version 5
HPC	High Performance Computing
HTTP	Hypertext Transfer Protocol

I/O	Input/Output
IDE	Integrated Development Environment
IP	Internet Protocol
IPF (IPFP)	Iterative Proportional Fitting Procedure
LDAP	Lightweight Directory Access Protocol
M	month
Moodle	a free and open-source software learning management system
OpenCOVER	Open COVISE Virtual Environment (an integral part of the COVISE visualization and simulation environment)
PBS	Portable Batch System (computer software that performs job scheduling)
PDF	Portable Document Format
PostgresSQL (Postgres)	an object-relational database with an emphasis on extensibility and standards compliance
Q&A software (FAQ Service)	a Web service that attempts to answer questions asked by users
RAID	Redundant Array of Independent Disks (a data storage virtualization technology)
RAM	Random-Access Memory
RDF	Resource Description Framework (a family of W3C specifications designed as a metadata model)
SCM	Software Configuration Management
SLURM (Slurm)	Simple Linux Utility for Resource Management (workload manager, job scheduler)
SM	Streaming Multiprocessor
SQL	Structured Query Language
SSH	Secure Shell (a cryptographic network protocol)
TB	Terabyte
VCS	Version Control System
VR	Virtual Reality
W3C	World Wide Web Consortium
WP	Work Package
XLS	Excel Binary File Format

1 Introduction

The portal is a single entry point for all CoeGSS stakeholders. It acts as a Web frontend to the CoeGSS services. The CoeGSS portal is a project with ambitious goals. It must offer convenient and powerful services to a broad category of users such as GSS experts searching for a relevant GSS knowledge bases, people who are interested in studying HPC and GSS, modellers from the GSS community, and so on. In addition, it aims to provide user-friendly interfaces to the whole software stack that potential portal users from the GSS community might need. The list of CoeGSS software includes data management components, tools for generating synthetic populations and synthetic networks, ABMS frameworks, model repositories, instruments for data analytics and data visualization. All of these must be evaluated and integrated into the portal as separate complementary services.

According to the description of actions, in this document we discuss achievements in implementing services specified in the deliverable D3.2 [1] as far as they are realized to be integrated in release 2 of the portal and the overall Centre's workflow. We intentionally leave the issues related to the DSLs for assembling GSS simulations beyond the scope of this document since this area since the scope of this deliverable is focused on direct services available via the portal.

This document is an essential part of documenting WP3 – ‘Methods and Tools for GSS on HPC’. It is a first report on the integration of the services into the CoeGSS portal. This report is a living document and the release at project M18. The document will evolve during the three project years. A next release of the document will be delivered as D3.6 in M28.

The remainder of this text is structured as follows. Section 2 contains a general workflow description, as well as a bird's-eye view on the structure of the portal and state of its development. In Section 3, we document the portal components which already operate and can be examined by the registered users. In Section 4, we describe trial interfaces to the software components which should be integrated as new services in release 2 of the portal. In Section 5, we highlight ideas and plans regarding integration of the software components which are still under active development themselves, and, thus, cannot be integrated into the portal at this step. In Sections 3-5, we do not only emphasize the implementation of interfaces for the services and present a brief overview of the tools, but also discuss in details potential use cases for these services. Finally, we draw conclusions in Section 6.

2 Current state of the portal

Within this section of the document, the status of the CoeGSS portal is briefly highlighted in order to set the scene for this document. It needs to be mentioned that deliverables D5.1 [10], D5.2 [11], D5.9 [12], and D5.10 [13] present all relevant information with respect to the portal concept and deployment, so that only a summary of those documents is included to avoid the duplication of information.

The initial CoeGSS Portal has been deployed in month M9 of the project's runtime and has been constantly improved in terms of management and failure prevention. However, no additional functionality has been made available to the users in the meantime – new features are planned for the second release in M20 that incorporate extensions, changes and improvements from WP3 and WP5. Although WP5 still coordinates and steers the overall development of the portal, the contributions of WP3 form the baseline for a successful inclusion of required Global Systems Sciences functionality.

So far, the CoeGSS portal consists of the following components:

- **Frontend service**
A single user entry point for the direct end users of the available services. For extended functionality, such as single sign-on to the services or a direct contact point to the CoeGSS partners, registration and authentication within the frontend are mandatory.
- **Data management service**
The CoeGSS Portal includes a data management instance that is based on the state-of-the-art Comprehensive Knowledge Archive network (CKAN). Data access, operations as well as manipulation and visualization of data are directly supported via the web interface.
- **Training organization service**
In order to organize the training material of CoeGSS, a dedicated instance is available. Like for CKAN, a well-adopted software package has been used and customized: Moodle.
- **User support service**
For managing user requests, a dedicated support framework based on the AskBot software package has been installed and configured. Like for the other services, AskBot is directly available via the frontend.

All services above are directly available to the users. For management purposes, the portal utilizes a couple of other services:

- **Infrastructure management**
Network services such as IP address management or network time protocol are offered in an entirely separated infrastructure environment. Furthermore, monitoring of the portal services is managed via this instance as well.
- **Software testing**

As stated in the deliverables of WP5, software testing was not the main focus during the first phase of the project. Nevertheless, integration testing as well as rudimentary software tests will be carried out during the second phase of the project.

- **Lightweight Directory Access Protocol (LDAP)**

For enabling unified user authentication, an LDAP server has been installed and configured. All portal services rely on this service in order to support user authentication as well as registration.

In summary, a well-performing portal, which follows a modular approach that enables the integration of extensions, has been developed and deployed. This concept is leveraged within this document in order to introduce more complex, but user-friendly and innovative functionalities.

As highlighted in the deliverables of WP5, the current version of the portal is available via the following link: <http://portal.coegss.hlr.de> .

3 Documentation of operating components

This section documents modules of the portal which already operate and might be examined by potential users. These modules were identified as required for integration in D5.9 [12].

3.1 Frontend

3.1.1 Purpose, Design, and Implementation

The main **purpose** of the frontend component is to provide a single point of access and to serve as the delegate for authorization/authentication for all other portal components implemented in the context of CoeGSS.

In order to achieve this purpose, LDAP is selected to be the central database for non-component-specific user data and authentication. Once the user registers to the CoeGSS portal, a user entry is created in the LDAP server; upon logging in, the user can access all portal components without the need of specific login procedures.

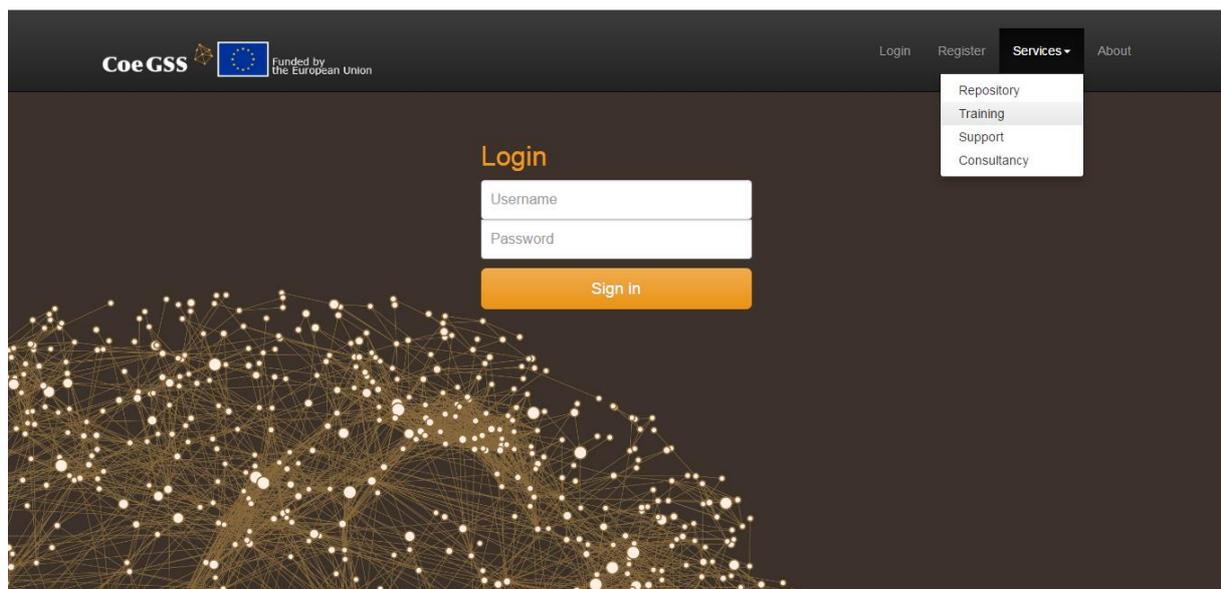


Figure 1 – Web-interface of the portal's frontend

The frontend basically consists of several web pages which allow users to register and login to the CoeGSS portal, access CoeGSS components such as CKAN, Moodle, AskBot, etc. (see Fig. 1).

In order to implement this web application, the Django Python based web framework was chosen. The current version of the frontend component requires Python 2.7, Django 1.9.6, PostgreSQL 9.4 and the following Python packages: django-registration-redux, python-ldap, django-auth-ldap.

3.1.2 Use Cases

Once an unauthenticated user accesses the CoeGSS frontend he/she will be redirected to the login page. At this point the user can log in, register, or access general information about the CoE. **Unregistered users** are allowed to access the services offered by CoeGSS with a limited functionality. These limitations are explained in detail in the following sections dedicated to each service.

Once the user logs in or registers to the portal, he/she will obtain the status of **regular user** for all provided services; in order to be granted a higher authorization level, the related service administrator has to be contacted.

3.2 CKAN

3.2.1 Purpose, Design, and Implementation

The CKAN is a web-based open source data management system that is mainly used by public institutions seeking to share their data with the general public¹. Nowadays, CKAN is the common place to search for open knowledge resources as well as register own. The CKAN is an important part of the entire CoeGSS workflow. It introduces data into the CoeGSS system and makes them available for all project users as well as for all applications implemented on the HPC systems.

In a nutshell, 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, 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 requirements identified in D4.1 [3].

In order to fully meet the requirements, improve the user experience and functionality, a number of **CKAN extensions** have been deployed:

- *Datapusher*: Whenever a structured resource is added to a dataset, this extension downloads the file (CSV or XLS), parses the data and pushes the data to Datastore.
- *Datastore*: Provides an ad-hoc database for structured CKAN resources which allows automatic data previews on the resource page. Adds search filter and update operations for the data without having to download, edit and upload the whole resource.
- *LDAP*: Allows LDAP authentication for CKAN.

¹ <http://docs.ckan.org/en/latest/user-guide.html#what-is-ckan>

- *Disqus*: Allows users to publish comments about datasets.
- *DCAT*: This extension provides plugins that allow CKAN to expose and consume metadata from other catalogs using RDF documents serialized using DCAT.
- *Harvester*: Provides a command line interface and adds a web user interface to CKAN for managing harvesting other CKAN instance datasets. By using either of these interfaces, a harvesting source can be set and the extension creates processes which download all the publicly available datasets from the source and adds them to the CKAN instance at CoeGSS. We refer to Section 4.1 for further information about the data harvester.
- *DREL*: This extension is being implemented in the context of CoeGSS. The purpose of the extension is to create and manage relationships between datasets such as; parent-child, dependency and derivation.
- *CoeGSS Theme*: This extension is created in order to modify the default CKAN look-and-feel to be in line with the CoeGSS visual style. CKAN encourages creating extensions for visual changes for the sake of maintainability and stability.

The CKAN instance deployed on the CoeGSS servers can be accessed by selecting the *Repository* menu item on the frontend's *Services* menu or directly at <https://ckan.coegss.hlrs.de>.

3.2.2 Use Cases

There are two main user roles in CKAN: system administrator and regular user. The CKAN **system administrator** is always allowed to perform any functionality provided by CKAN. A **regular user** has to be a part of an organization or a group in order to be able to gain more rights (excluding the user's own profile management) than non-authenticated visitors. A **non-authenticated visitor** can view publicly available data on CKAN, search and view datasets and access the visualization tool. Any user registered to the CoeGSS portal is assigned the regular user role and will have all the rights of a regular user – plus the rights that come with the organization and/or group memberships. The latter are detailed in the paragraphs below.

Upon accessing CKAN, a user who is already registered to the CoeGSS portal, will be able to *view* all the publicly available *datasets* and all groups. In order to be able to create or edit content, the user has to be assigned to an organization. Therefore, a mechanism where the user can reach organization administrators to request an invitation or reach a system administrator to request a new organization to be created has to be implemented.

Once the organization assignment is done, the user will be able to *create*, *edit* or *delete datasets* belonging to that organization. Any user assigned to a group will be able to take the actions stated in Subsection 3.2.2.2, depending on the group role assigned.

Besides handling datasets, the users will also be able to *manage relationship links* between the datasets belonging to their organization.

3.2.2.1 *Organization membership*

In CKAN, each dataset has to belong to an organization and a dataset can only be owned by a single organization. In CKAN organizations control which user can see, create and update these datasets. A user can have one of these three roles in an organization: admin, editor, and member. An **organization admin** can:

- view the organization's private datasets;
- add new datasets to the organization;
- edit or delete any of the organization's datasets;
- make datasets public or private;
- add users to the organization, and choose whether to make the new user a member, editor or admin;
- change the role of any user in the organization, including other admin users;
- remove members, editors or other admins from the organization;
- edit the organization itself (for example: change the organization's title, description or image);
- delete the organization.

An **organization editor** can:

- view the organization's private datasets;
- add new datasets to the organization;
- edit or delete any of the organization's datasets.

An **organization member** can:

- view the organization's private datasets.

3.2.2.2 *Group membership*

Groups in CKAN work like controlled tags and allow categorizing datasets. A dataset can belong to arbitrary number of groups. There are two different roles in a group: admin and member. A **group admin** can:

- add datasets to the group or remove existing datasets;
- add or remove group members, and choose whether to make the new user a member or an admin.

A **group member** can:

- add datasets to the group or remove existing datasets.

3.3 Moodle

3.3.1 Purpose, Design, and Implementation

Initial portal design identifies the need for *providing training material* for both HPC and GSS domains since this combination of fields is rather new and it requires guidance for new end users aiming to adopt the technologies provided by the CoE.

Besides the must-have guides and courses, the **requirements** of a *training manager* also include workshops to allow students to experiment with what they learn, discussion boards within each course and a feedback mechanism in order to improve the published material.

Among the open source education platforms, Moodle was identified to be the one which fits CoeGSS best by covering all the requirements for the training manager, being highly customizable, and having a strong community behind it.

The component depends on LDAP for user authentication thus; any user who has already logged in to the CoeGSS portal would be able to access the training services without needing to log into Moodle.

No training material has been created in Moodle yet. Content for courses and workshops should be created and these training materials should be organized under appropriate categories in the next releases of the portal.

The Moodle instance will be enhanced with the following **plugins**:

- *Configurable Reports*: This Moodle plugin enables the content creators to produce course, user, category or timeline reports without requiring SQL knowledge. An example for a report would be the information of users and their activities in a specific course. The plugin also features filters, pagination, logic conditions and permissions, templates support and exporting reports to XLS format.
- *Certificate*: This plugin allows the creation of PDF certificates/diplomas for the students of the course which are completely customizable (borders, watermarks, seals, grade information etc.). It also implements a verification mechanism for the certifications which is useful when a supervisor or administrator wishes to verify that the printed certificate is valid for that student.
- *Questionnaire*: Allows the teachers to create a set of questions to get student feedback on a course, an activity, etc. The goal of this plugin is not to create a gradable item such as a quiz to assess the students, but to gather feedback data about the course to be analyzed.

The component has yet to be updated with the new look and feel to be in line with the one that has been defined for the CoE in general.

The Moodle instance deployed on the CoeGSS servers can be accessed by selecting the *Training* menu item on the frontend's *Services* menu or directly at <http://moodle.coegss.hlrs.de/>.

3.3.2 Use Cases

Moodle is available for all registered users but the non-authorized users are only able to view limited content; home page, list of courses, course calendar etc. There are several **roles** within Moodle one of which will be assigned to the users:

- *Course creator*: can create courses.
- *Teacher*: can manage and add content to courses.
- *Non-editing Teacher*: can grade in courses but cannot edit the courses.
- *Student*: can access and participate in courses.

Once the **course creator** creates a course and assigns a teacher to a course, the user with the teacher role will be able to plan and shape the course. After the course creation, it will be listed on the *navigation block* of every student's home page. The **non-editing teachers** will be responsible for grading and following course discussion boards.

Moodle allows role assignment for a particular context, in other words, roles are not globally defined. This allows a teacher to be enrolled to a class as a student or vice versa. By default, newly registered users will be assigned the role of **student**.

3.4 AskBot

3.4.1 Purpose, Design, and Implementation

The design of the portal includes features for interacting and providing information about the services for both registered users and unregistered visitors of the website. Interaction with registered users includes the implementation of a customer support system structured as a question and answer forum.

Requirements of the system include:

- a tagging system to provide flexible categorization;
- a voting system in order to mark the most useful information;
- filtering of all new issues submitted;
- filtering of all issues closed with an answer;
- filtering of all issues still open without an answer.

The free software product AskBot is the tool chosen to implement such a system.

The AskBot system interfaces to the CoeGSS portal through LDAP allowing users with a valid portal login to authenticate on AskBot in order to raise questions and interact with the CoeGSS support staff.

Categories need to be created in the AskBot system to match the services available on the portal. When new services are made available on the portal the list of categories in AskBot needs to be reviewed and updated.

The portal needs to include links to the AskBot system in those pages where access to the support system might be relevant.

The look and feel of the AskBot system has been customized using the provided customization options to match the general style of the CoeGSS website and portal. These customizations are packaged in a single CSS file that can be easily applied to new installations and that is carried forward between AskBot software updates.

The current implementation of the system is reachable at <http://support.coegss.hhrs.de/>

3.4.2 Use Cases

The system provides information about problems encountered by the users of the services through the CoeGSS portal. It is open to non-authenticated visitors to browse previous questions raised by the users and it requires authentication in order to raise new questions. Users authenticate with AskBot using the same credentials used to access the portal.

For **non-authenticated visitors**, the *existing questions* (either answered or being discussed) can be browsed freely in order to find the solution or to follow the relevant conversation which will hopefully lead to the resolution of the issue.

For **portal users**, the same credential valid for the CoeGSS Portal can be used to access the AskBot system in order to raise a support request. The system presents an input box marked as "*search or ask your question*" where the user starts entering the subject of his/her request. Existing questions matching the keywords entered will appear along with the option of creating a new question. The user can optionally *tag the question* giving an indication of the services impacted by the problem, support staff will add/remove tags according to the areas where the issue needs to be investigated. The user will come back to the portal to check the progress of his/her request and, once support staff provides an answer, the user will be able to *mark the provided solution as valid or submit more information* if the provided solution does not solve the problem.

Support staff will *identify new questions and escalate each question* as needed to receive information from the technical staff working on the impacted services. Support staff acts as a "**first line**" support and logs into the AskBot system with "support staff" permissions. AskBot provides a quick way to show the questions for which an answer has not been defined by selecting the "Unanswered" filter in the top menu. Support staff provides follow-up information and encourages the user to test the answer provided and *mark the answer as correct or describe why the answer does not solve* the problem in order to reach a satisfactory solution. Support staff can also *escalate the issue* to technical points of contact for the service the problem is related to. Each person acting as a technical point of contact acts as a "**second line**" support and has the specific expertise to troubleshoot one of the specific services

provided by the portal. Their task is to provide information to the first-line support staff or take directly the responsibility to deal with the user through the AskBot system until a solution is reached.

4 Software integration

This section describes trial interfaces to the software components which must be integrated into release 2 of the portal. This software was identified as an essential part of the general CoeGSS workflow in D3.2 [1].

4.1 CKAN data harvester

4.1.1 Overview

The CKAN harvester is designed to import data from outside sources to the CKAN server. We implemented it as a CKAN plugin. Information about the data location can be provided as:

- a link to a remote location;
- the location of a file located at a local disc.

In order to transfer data from the CKAN server to an HPC server a special application is used (CKAN client).

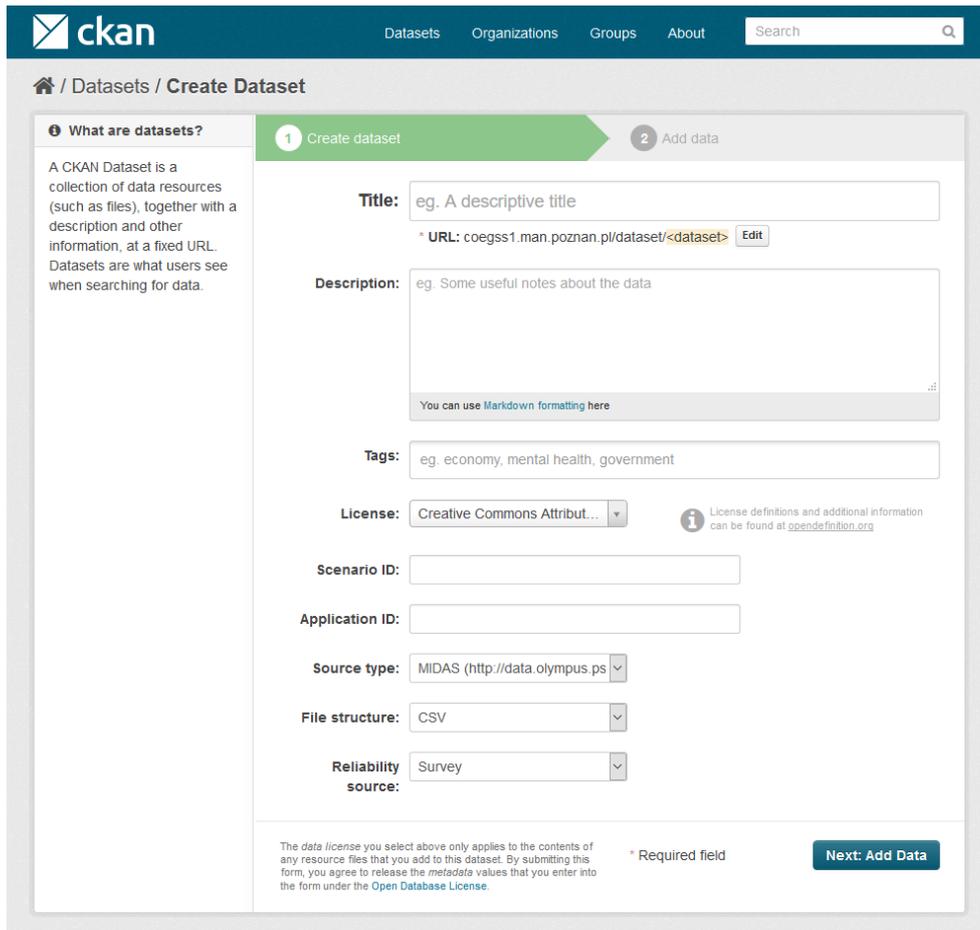
The harvester is currently capable of handling the following file types: CSV, TXT, XLS. Handling means the process of automatic data parsing and conversion from the format used in a particular file to a tabular form used by the database. This conversion provides a lot of additional functionality from the CKAN service like:

- data can be explored directly in the portal;
- in case spatial data exist (columns longitude, latitude), a map is generated which represents these data (e.g., shows pins representing particular agents);
- data can be extracted from the database in partial form (not only entire files).

All other file types are stored in the form they were delivered without any converting procedures.

4.1.2 Implementation and Web interface

An intuitive web interface allows dataset publishers and curators to easily register, update and refine datasets in a distributed authorisation model called *Organizations*. Organizations allow each publisher to have their own dataset entry and approval process with numerous members. This means responsibility can be distributed and authorization access managed by each organization admins instead of centrally.



The screenshot shows the CKAN 'Create Dataset' form. The header includes the CKAN logo and navigation links for Datasets, Organizations, Groups, and About, along with a search bar. The breadcrumb trail is 'Home / Datasets / Create Dataset'. The form is divided into two steps: '1 Create dataset' (active) and '2 Add data'. A sidebar on the left provides information about datasets. The main form area contains the following fields:

- Title:** A text input field with the placeholder 'eg. A descriptive title'. Below it is a URL field with the placeholder 'eg. coegss1.man.poznan.pl/dataset/<dataset>' and an 'Edit' button.
- Description:** A large text area with the placeholder 'eg. Some useful notes about the data'. A note below indicates 'You can use Markdown formatting here'.
- Tags:** A text input field with the placeholder 'eg. economy, mental health, government'.
- License:** A dropdown menu currently set to 'Creative Commons Attribut...'. An information icon and link are provided for license details.
- Scenario ID:** An empty text input field.
- Application ID:** An empty text input field.
- Source type:** A dropdown menu currently set to 'MIDAS (http://data.olympus.ps)'.
- File structure:** A dropdown menu currently set to 'CSV'.
- Reliability source:** A dropdown menu currently set to 'Survey'.

At the bottom, there is a note about license application and a 'Next: Add Data' button. A '* Required field' label is also present.

Figure 2 – Submission form: 1st stage – providing general information

The CKAN data service uses two PostgreSQL **databases**: *ckan_default* and *datastore_default*. The *ckan_default* database contains information about datasets, users, metadata. The second database *datastore_default* has several tables with imported data – one CSV, TXT or XLS file in one table.

A common way to **upload** data to the platform CKAN is to use the DataStore API. The second method is to pass a link to an external source. This method is less recommended due to the restrictions (e.g., limitations in file size).

Depending on the number of records, the input data parsing procedure takes from a few to several minutes or more. Once transfer of the data into the CKAN DataStore is finished the file from an external source can be deleted, the data remains in the database.

The CKAN *DataStore* extension provides an ad hoc database for storage of structured data from CKAN resources. Data can be pulled out of resource files and stored in the DataStore. The DataStore (like CKAN) requires the PostgreSQL 9.2 (or later) database. This was released in 2012 and is widely available.

The **submission** form, specifically tailored for CoeGSS needs, consists of fields necessary to describe data in a way which allows further searching and exploring. The submission procedure consists of two *stages*:

- providing general information related to the imported data (see Fig. 2);

- providing a localization info where data come from (see Fig. 3).

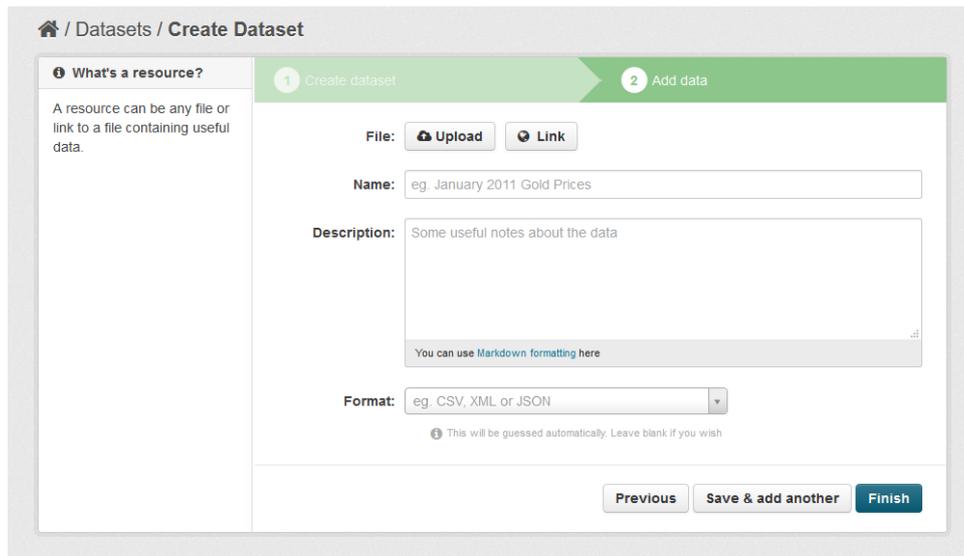


Figure 3 – Submission form: 2nd stage – providing a localization information

Once the importing procedure is finished and data are converted into tabular form, they can be easily explored directly in the CKAN portal. CKAN can **display** submitted data in different ways. E.g., when looking at a synthetic population, a map is generated presenting the location of the agents and (after click) all related metadata in a specific popup window (see Fig. 4).

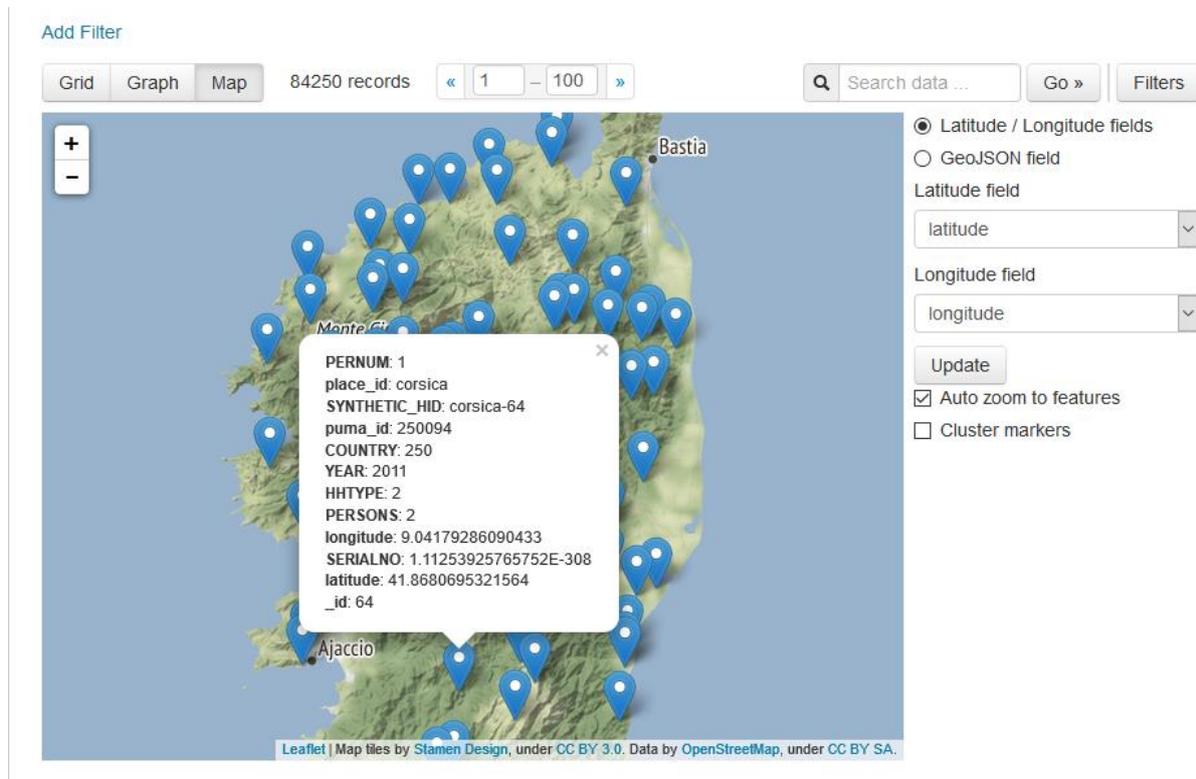


Figure 4 – Data presented in a map form

4.2 SynPop and synthetic network tool Web GUI

4.2.1 Requirements and Implementation

The synthetic population generation and synthetic network reconstruction components will be accessed from either the portal's job or workflow management system. In order to generate a synthetic population or a synthetic network, a generation job will need to be specified. The intended users for the component will be model developers with programming knowledge. In the future, an alternative, simpler interface might be provided for modellers without programming knowledge. However, this depends on whether it is possible to identify a simpler way of describing the job that is able to handle less demanding situations.

The **requirements** include:

- being able to specify a job that generates a synthetic population using common methods such as IPF and uniform sampling;
- being able to use marginal data distributions (of one or more dimensions) and optionally a microsample;
- in addition to the synthetic population, the component generates a simple signature that indicates the provenance of the data and the method used for generating the synthetic population;
- the component supports generating a synthetic population based on older data if some of the input files exist in versions from different dates;
- optionally, the description of the job supports simple transformations of the data, such as filtering or conversions;
- optionally, the description of the job allows for using components defined in other descriptions to allow for sharing the job implementation code without copy-and-paste cloning.

The **job specification** will have the following parts:

- the vintage parameter;
- input data;
- output data;
- job description.

The *vintage parameter* is either a past timestamp or the label 'latest', which is equivalent to the current timestamp. The vintage parameter allows selecting older versions of the input files.

The *input data* identifies the datasets that are to be used by the generation procedure. The datasets are described by CKAN resource IDs (see Section 3.2), and optionally with the paths that identify particular tables inside of HDF5 files. For every input file, the version with the latest timestamp that is not more recent than the vintage date is selected.

The *output data* identifies the path to the CKAN dataset where the resources containing the result data will be created, and the names of the resources.

The *job description* specifies the processing job that builds the synthetic population. A job is a sequence of operations, such as an IPF or sampling run, which takes inputs (either external or intermediate files) and creates outputs (either the final or intermediate files).

Optionally, the job description can include simple transform or filtering operations. In addition, it may refer to compound operations that are defined in other job descriptions. A compound operation is similar to an ordinary job description, but contains formal parameters instead of the input files.

Job specification, defined this way, can be compiled to yield an *execution plan* if the compilation is successful, or yield an error message otherwise. Among other things, the compilation process checks if the files referenced by the job description exist, and whether they contain data of the required form. A successfully compiled execution plan will then be able to be executed by the job execution component.

Generating **synthetic networks** will be part of the process for generating synthetic populations. To this aim, the job description will contain additional algorithms for network reconstruction. Network reconstruction methods generate viable candidates for the network under consideration, satisfying the constraints represented by the available information about the original network. Essentially, network reconstruction methods fall in two classes: generative algorithms and ensemble methods. *Generative algorithms* create a synthetic network dynamically, starting with an empty network, with the aim of reproducing a set of observed quantities (i.e., the constraints represented by the available information on the real network). Being developed in the Exponential Random Graphs framework, *ensemble methods* define link-specific probability coefficients (whose computation is based on the constraints) and, then, generate synthetic networks by sampling from the induced probability distribution. Ensemble methods are not always the fastest ones, but permit to take control of fluctuations (which is impossible in generative methods) and other higher moments of the ensemble probability distribution; moreover, in most cases they are ergodic (i.e. explore the phase space homogeneously), a characteristic which is not easily provable in the case of generative methods. The crucial difference among the two different sets of methods lies in the different kind of insight provided on the mechanism driving the network formation. While generative methods need to define microscopical rules (which, somehow, model the behaviour of nodes/agents), ensemble methods do not, being more suitable to select the set of configurations showing the largest compatibility with the known information.

Input files required for network reconstruction may vary a lot:e.g., they may be the sequence of the degrees of the vertices, the parameters of the degree distribution, a biadjacency matrix related to the monopartite network (whose nodes could be the agents of the synthetic population) to be reconstructed. Depending on the input, the approach for the reconstruction will change; anyway, on the basis of the previous arguments, ensemble methods will be preferred.

4.2.2 Access and Use Cases

The synthetic population and synthetic network generation components will be accessible to **logged-in users** with specific LDAP access rights. The user will be billed for the generation job according to the common rules for billing.

4.3 Visualization tools

Within the project, the focus of the Visualization Task is to develop and to provide remote and immersive visualization services to consortium partners as well as to CoeGSS users. These services will be integrated into the CoeGSS portal regarding the defined workflow as described in D3.2 [1]. These services will provide access to HPC as well as Visualization resources integrated in a seamless manner in order to create “Immersive Analytics Environments” for huge statistical and multidimensional datasets.

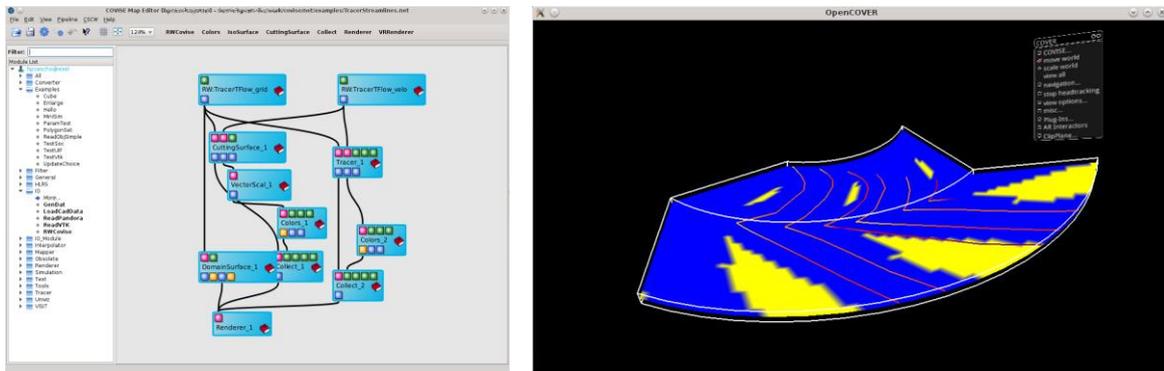


Figure 5 – COVISE Map Editor and Renderer

As a result of the software survey, reported in D3.1 [2], the Collaborative Visualization and Simulation Environment (COVISE) seems to be a promising candidate to fulfill needed requirements, to handle expected data volume, as well as to be integrated into an HPC environment.

This subsection will describe current development on interfaces to access datasets with respect to the proposed workflow definition as well as modules to read and process datasets for visualization which shall be integrated as new services in release 2 of the portal.

4.3.1 Requirements and Implementation

Most important requirements regarding visualization have been summarized within D3.1 [2] and D4.2 [4]. **Requirements** can be classified roughly into four major **categories**:

- *interfacing data storage*

GIS data is planned to be stored or referenced either locally on a desktop machine or remotely, either directly on HPC storage systems or by using portal services such as CKAN (see Section 3.2). Downloading of stored or referenced data to the COVISE environment running either on a local machine or on an HPC machine via a secured network connection is required. Secure protocols and mechanism have to be defined and implemented to enable CKAN clients or COVISE respectively for the CKAN server.

- *converting / processing data for visualization purposes*

First conversion and preparation steps might be performed within the remote CKAN system or the portal respectively (e.g., check data integrity and completeness). The visualization tool should offer additional modules to process/prepare the data automatically (e.g. data conversion, matching, sub sampling, compositing) once the data flow is set up and configured due to user demands.

- *interactive visualization and user interaction*

The software framework should support interactive rendering to support immersive visualization environments like CAVEs or remote visualization on HPC systems for instance. The user should be able to interact with the data to get different views on the data or to compare datasets of runs for different parameters for instance as well as to interact with data to be able to choose different datasets or different selection/subsamples of the data.

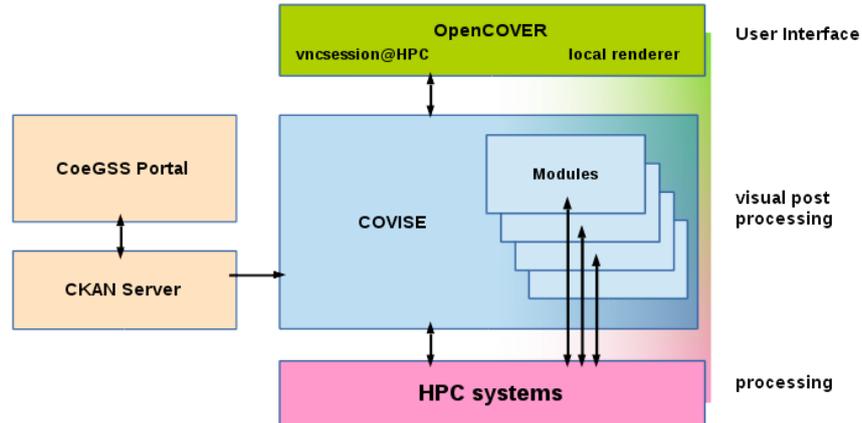
- *reporting*

The software should support reporting mechanisms like generating video files or screenshots linked to users meta data.

Requirements as stated by deliverables and constantly collected from consortium partners as well as users are summarized, refined, prioritized and tracked in a database for further software development.

With the second release of the portal, **design** and **interfaces** between software environments and frameworks have been refined based on the design reported with D3.2 [1].

Figure 6 – COVISE/OpenCOVER Integration



Derived from major requirements work package subtasks in visualization can be subdivided into the following sections:

- *UI and software deployment*

This subtask focuses on software availability on HPC systems to support remote rendering services and deployment of software for local usage by CoeGSS users.

The rendering engine OpenCOVER provides standard 3D navigation and interaction as expected from 3D visualization tools. Furthermore, OpenCOVER includes a huge variety of plug-ins to handle visualized data interactively. Currently all UI requirements given by users or described by deliverables are being met. The software is available on the pre- and post-processing servers on HazelHen at HLRS which are dedicated for data pre- and post-processing including visualization. Also planned is the installation on PSNC’s HPC computer system Eagle. COVISE/OpenCOVER are available open source on GitHub and as binary download for Microsoft Windows, Linux and MacOS X. While the software installation including CoeGSS modules on HPC systems is updated monthly at the latest, the GitHub repository is under continuous development.

- *Portal/CKAN interface*

This subtask focuses on design and implementation of the COVISE interface to CKAN using the CKAN API and CoeGSS services.

There are different possibilities to access data hosted or referenced by CKAN from COVISE. This can be done by using the CKAN API which is mainly based on HTTP or by using references to access the data directly via secure file transfer from HPC file systems for instance. In order to be able to handle huge datasets, currently different approaches are being implemented, tested, and benchmarked. Especially a seamless integration into the CoeGSS workflow, as well as the integration in HPC operation is a major issue.

- *Modules*

This subtask focuses on module and plugin development regarding user requirements and demands, as well as needed data formats. These modules can be either run on HPC/remote systems or local clients depending on referred workflow.

Modules and plugins are usually implemented to integrate user specific demands or external tools and offer new processing or reading capabilities for instance. In addition to basic functionalities of COVISE and OpenCOVER around 100 plugins and 400 modules are currently available on GitHub.

The *COVISE module ReadPandora* is a reading module based on the COVISE HDF5 reader, which is available already. The HDF5 reader makes use of the libHDF5 library or the HDF5 C++ API description respectively. As reported in Deliverable 4.2, HDF5 is proposed to become the standard format for CoeGSS data. The ReadPandora module currently enables the user to select an HDF5 file and the parameters from within the file. With executing the module, the module reads the data into COVISE and offers data output as polygon mesh, which can be read by the renderer directly.

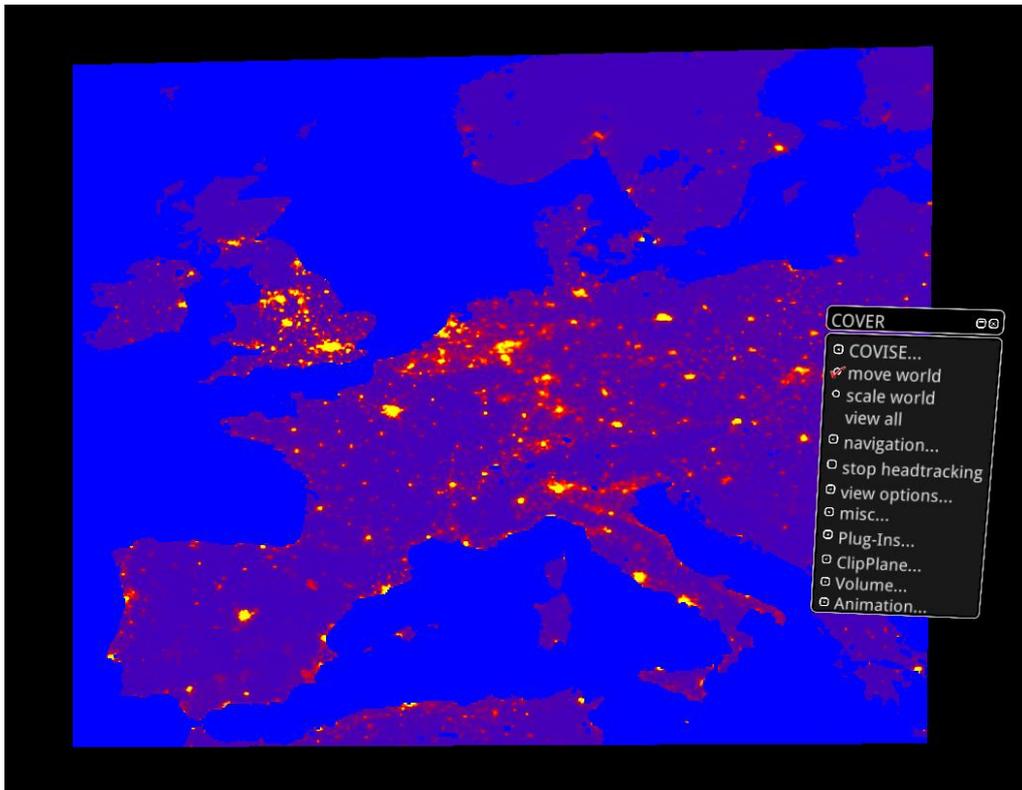


Figure 7 – ReadPandora sample dataset

Storing the data as polygon meshes uses a huge amount of memory but enables quick setup of a visualization and enables further processing, which might be an advantage at this stage of the project. As soon as the CoeGSS workflow has been specified in more detail and the CoeGSS-ABM framework becomes ready, the CoeGSS data format can be specified.

Consequently, the data will be read into a more suitable and memory optimized COVISE container then.



Figure 8 – Showcase of PPT link from within a CAVE

The *OpenCOVER plugin PPTAddIn* was implemented to support the documentation process while using VR environments for instance. While using a CAVE for example, users usually have no laptop or tablet computer at their hands to write down comments or annotations. In fact, the user can add annotation markers to the scene or make screenshots, but these are usually stored on a file system with no context to the VR session. The PPTAddIn plugin enables the user to define and to link screenshots instantly into a Microsoft PowerPoint presentation for example, to track progress of an investigation or data analysis.

4.3.2 Access and Use Cases

An important current task within the CoeGSS project is to provide access to the CoeGSS visualization software to support the CoeGSS consortium developing further reading and processing modules as well as to test, refine requirements, and to identify additional needs.

COVISE/OpenCOVER has been installed on the pre- and post-processing servers on Hazelhen at HLRS. These machines are dedicated for memory I/O intensive workload like visual post processing for instance. There are 14 servers available, which are configured as follows:

3 Cray CS300	each with 4x Intel(R) Xeon(R) CPU E5-4620 v2 @ 2.60GHz (Ivy Bridge), 32 cores, 512 GB DDR3 Memory (PC3-14900R), 7,1TB scratch disk space (4x ~2TB RAID0), NVidia Quadro K6000 (12 GB GDDR5), single job usage
5 Cray CS300	each with

	2x Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz, 16 cores, 256GB DDR3 Memory (PC3-14900R), 3,6TB scratch disk space (2x ~1,8TB), NVidia Quadro K5000 (4 GB GDDR5), single job usage
3 Supermicro Superserver	each with 4x Intel Xeon X7550 (Nehalem EX OctCore), 2.00GHz (4*8=32 Cores for 32*2=64 HyperThreads) 128GB RAM, 5,5TB scratch disk space (10x ~600GB), NVidia Quadro 6000 (GF100 Fermi) GPU, 14 SM, 448 Cuda Cores, 6 GB GDDR5 RAM (384bit Interface with 144 GB/s), single job usage
1 Supermicro Superserver	with 8x Intel Xeon X7550 (Nehalem EX OctCore), 2.00GHz (4*8=32 Cores for 32*2=64 HyperThreads) 1TB RAM, 6,6TB scratch disk space (14x ~600GB), NVidia Quadro 6000 (GF100 Fermi) GPU, 14 SM, 448 Cuda Cores, 6 GB GDDR5 RAM (384bit Interface with 144 GB/s), multi job usage
2 Cray CS300	each with 4x Intel(R) Xeon(R) CPU E5-4620 v2 @ 2.60GHz (Ivy Bridge), 32 cores, 1536 GB DDR3 Memory (PC3-14900R), 15 TB scratch disk space (4x ~4TB RAID0), NVidia Quadro K6000 (12 GB GDDR5), multi job usage

Table 1 – Pre- and post-processing servers on Hazelhen@HLRS

As described in D5.1 [10], at HLRS, there is a completely manual process to get access to the systems. In particular, two forms have to be filled in, signed by the user and sent to the HLRS premises. In order to simplify access to the documents for the consortium partners, both have been uploaded to the project subversion system at the following location: [https://scm.projects.hlrs.de/svn/coegss/trunk/Resource Access/](https://scm.projects.hlrs.de/svn/coegss/trunk/Resource%20Access/). Access procedures and instructions to configure and run specific software tools are described on the corresponding wiki-portal: https://wickie.hlrs.de/platforms/index.php/Cray_XC40.

As stated in 4.3.1, installation of COVISE/OpenCOVER including the CoeGSS visualization modules and plugins on PSNCs HPC computer system Eagle is planned.

5 Software outlook

This section highlights ideas and plans regarding integration of the software components into further releases of the portal. As this section covers tools which are either only partially specified (data analytics tools) or still under development (ABMS frameworks), this section only gives an insight on the directions we wish to move. It provides preliminary versions of the interfaces between the tools and the portal, whereas the final version will be specified in the further version of this live document which is due in M28.

5.1 ABMS interface

5.1.1 Requirements and Implementation

Agent-based modelling and simulation is an essential part of the portal's workflow. Next releases of the portal must offer services **aiming** to facilitate the process of preparing and simulating agent based models on the HPC platforms of the supercomputing centres.

In order to support GSS modellers with an easy-to-use domain specific programming environment for simulating evolution of ABMs, the portal incorporates access to *agent-based modelling frameworks*. During the first year of the project, a number of ABMS frameworks has been studied and analyzed. Two of them – RepastHPC and Pandora – have demonstrated the greatest potential as the tools of choice for HPC-driven agent-based modelling [1,2]. Nevertheless, none of these tools completely satisfies all requirements imposed by the pilots on the ABMS frameworks for GSS modellers (see [3–6]). Moreover, we came to the conclusion that it is impractical to invest efforts on updating these tools for a number of reasons. In particular, Pandora uses poor rigid design solutions, and, thus, cannot be easily extended with the new functionality [5–7]. In contrast, RepastHPC has a rather elaborate design, but it provides an overcomplicated and verbose API which requires deep expertise in C++, and, thus, hardly can attract a broad community of GSS modellers [5,6,8]. This discovery stimulated us to initiate development of a new ABMS framework. In this framework, apart from support of pilot requirements, we aim to implement an elaborate design with high level of extensibility, and to equip our end-users with a straightforward and simple to use API. The new framework must address critical HPC related issues which were left beyond the scope of other ABMS frameworks such as adaptive load balancing, efficient work with spatially irregular agents' environments, etc. As soon as we finish with the first release of the new framework, it will be integrated into the portal along with Pandora and RepastHPC suites. All three frameworks are written in C++, but require different levels of C++ expertise from the end users.

First and foremost, ABMS frameworks integration assumes development of the Web interface for specifying, running, and tracking the simulation jobs. Within this interface, the portal should **provide** the following **functionality** for authorized users:

- uploading of model sources to the portal and further to the target platforms;

- compiling of models on the target platforms;
- specification of input data (rasters, social graphs, serialized synthetic populations, information about agents' environment) and simulation parameters (simulation timeframe and number of steps, frequency of data output, etc.);
- convenient switching between the target HPC platforms of the supercomputing centres;
- tracking of submitted simulation jobs;
- downloading of simulation results and forwarding them to the post-processors (visualization and data analytics services).

Apart from support of these basic requirements, the portal might offer other facilities such as generators of ABM projects for different IDEs, synchronization with ABM repositories, wizards for ABM projects, etc. Nevertheless, the latter requirements are a subject of further discussion.

In order to match implementation of the ABMS Web interface with other portal services, it has to be written as a Django Web application in Python.

5.1.2 Access and Use Cases

The portal should allow to simulate models only for authorized users. One can distinguish two phases in interaction between the end user and the ABMS interface of the portal: preparing an ABMS executable and running simulations.

5.1.2.1 Model preparation phase

The model preparation phase is related to implementing a model with the help of the ABMS framework of choice. There are two main user roles with regard to this phase: system administrator and modeller. **Modellers** prepare the sources of the models, whereas **administrators** review the codes of uncertified modellers and approve for the further simulation. *Code review* should help to avoid launching unreviewed codes on the HPC systems of the supercomputing centres. We consider as **certified modellers** only those users who signed the required legal papers with the HPC centres or were granted with remote access to HPC platforms of PSNC and HLRS in some other way. The others automatically fall into the category of **uncertified modellers**.

The portal should offer support for both *off-line desktop developments* and *developments with remote access* to the HPC resources. In the first case, the modeller must implement and verify his model locally, and afterwards upload the sources to the portal for further review if required. This option is valid for all authorized users, but it assumes that the modeller has a desktop version of Pandora, RepastHPC or the CoeGSS-ABMS framework, respectively. In this case, sources, approved after review, will be compiled against the chosen framework and with the options specified by the modeller, which validates the model for the simulation phase.

We plan to support the users with remote access to the HPC resources via SSH by a tool which generates empty IDE projects on the target supercomputers for the chosen ABMS framework. This will help the end users to automate the steps of setting up development environment and preparing build scripts. The initial version of the tool supports the Sublime Text 3 and the Eclipse CDT projects. In both cases, we assume that the user installed remote access plugins in his/her IDE – rsub for Sublime and RSE (Remote System Explorer) for Eclipse. Later on, this list might be extended by other popular editors and IDEs like Emacs, Vim, QtCreator, etc.

We also discuss an alternative approach to organize the model development process according to which the modellers can interact with the portal via registered *version control repositories*. This might enable collaborative work of the modellers on the same ABM sources and simplify a lot the code review process with the tools like Gerrit, side-by-side diff view in GitLab, etc. In addition, this approach gives an opportunity to provide some automated SCM services such as build management, etc. Finally, it closely correlates with one of the CoeGSS goals which aims on “provisioning of code repositories and building community around the GSS applications” [9]. While the benefits of this approach are significant and evident, it requires substantial efforts in its implementation. Therefore, we do not expect to implement it in the near future and will likely omit this feature in the final version of the portal.

5.1.2.2 *Simulation phase*

In the simulation phase, the user specifies and submits the simulation job. The portal has to provide the same job specification and submission interface for **all authenticated users** with proper rights and access to ABM executables. This interface requests from the user a number of parameters such as target supercomputer, model name, configuration parameters (conveyed either within configuration files or as command-line arguments), input files, output paths, etc. As soon as the job is completely specified and submitted via the ABMS Web interface, the portal creates the workspace for model execution, generates the batch script for running a simulation job (either PBS or Slurm), and puts this script into the queue of the job scheduler on the target HPC system. In response, the user should receive notifications from the portal when the job execution is started and finished. Once the user is notified that the job is completed, he/she can either store/download results or proceed to data analytics and visualization.

In general, the Web interface for the simulation phase is similar to the Web interface for the SynPop tool with regard to its implementation and functionality since both interfaces serve the same purpose – running executables on the HPC systems of the supercomputing centres. Therefore, we refer to Section 4.2 for further detail.

5.2 Data analytics interface

5.2.1 Requirements and Implementation

Data analytics is a pervasive part of the pipeline of HPC for GSS in the approach based on synthetic information systems.

When the data needed for creating the synthetic population is gathered, different data sources will be used, leading to a need for pre-processing, cleaning and indexing. There are also questions related to processing the output data and dealing with the uncertainties that originate in the input data and in modelling assumptions.

To be able to extract the information of interest to the user, data analytics that combines user friendliness with advanced analysis is needed.

Examples of data analysis that will be **required** are:

- filtering of data;
- merging of big datasets from different sources;
- estimation of missing values;
- sensitivity analysis;
- clustering of data.

The tool chosen to combine the requirements of user friendliness and high performance is Apache Spark, that is an open source cluster computing framework, based on the Hadoop ecosystem and originally developed at the University of California, Berkeley's AMPLab.

Spark is designed in such a way that the users do not have to think how to parallelize their code themselves – as long as they write their code using the Spark paradigm the parallelization will be handled automatically, under the hood by Spark. This resonates very well with our aim that our service shall provide its users all the power and possibilities of high performance computing without requiring them to be computer scientists.

As Spark is Java based, jobs submitted to it can be written in Java or Scala, but it also has support for Python and R for the user who prefers to use one of those languages. There is also support for data frames and SQL.

On top of this, Spark comes preloaded with parallel versions of most of the commonly used data analytics algorithms, e.g., machine learning algorithms such as principal component analysis, logistic regression, k-means clustering, and graph algorithms such as triangle counting and PageRank. Spark is also preloaded with algorithms for streaming. This is probably nothing that will be of use at the moment, but as time passes and the service provided by CoeGSS grows, it is plausible that streaming of, e.g., twitter data will be incorporated in the product. Having a tool which already provides this will of course make things easier later on.

The initial demands for data analytics within CoeGSS are modest, focusing on filtering and merging of datasets, something that can easily be handled both by Spark and by less sophisticated data analytics tools. However, as the simulations become more advanced, we expect their needs to become more demanding, requiring us to use the more sophisticated algorithms of Spark.

5.2.2 Access and Use Cases

Spark is part of Cray's analytics platform Urika-GX and available at HLRS. It will be possible to give each **end user** access to an account and run their own Spark jobs. Another solution would be that a number of commonly used analysis jobs are preconfigured and possible to select by the end user, so that no actual coding would be needed.

For the more **advanced users** that require a previously not used analysis, this can either be arranged in such a way that one of the centre's experts writes the code, or if they are confident enough, the end user can write the Spark code themselves.

6 Summary

The current release of the portal includes four fully operating components: a frontend which serves as a single point of access for all users; a data management component (CKAN) which facilitates handling of datasets; a training organization component (Moodle) which provides training services; and a user support component (AskBot) which solves issues related to support of the users in a Q&A manner. In addition to these components, we plan to integrate into release 2 of the portal new services which must manage interaction of the users with the software installed on the HPC systems of our supercomputing centres such as CKAN harvester, synthetic population and network generation tools, and visualization tools (COVISE). This software covers a significant part of the general portal workflow as it was defined in D3.2.

On the other hand, we postpone integration of such relevant pieces of the general portal's workflow as ABMS and data analytics software to the next releases of the portal. The reasons are the active software development process of the ABMS software and obvious gaps in the specification of requirements to the data analytics part. Nevertheless, in this document we present our preliminary thoughts about integration of these components. The next release of this deliverable (to be released at M28) will introduce advances in the integration of these software components, as well as changes related to the integrated components.

References

- [1] P. Jansson, M. Lawenda, E. Richter, U. Woessner, C. Ionescu, M. Pałka, R. Schneider, D. Dubhashi, M. Tizzoni, D. Paolotti, S. Fürst, and M. Edwards. *D3.2 – Specification of new method, tools and mechanisms proposed*. Technical report, CoeGSS, 2016.
- [2] E. Richter, W. Schotte, C. Ionescu, R. Schneider, D. Dubhashi, and M. Lawenda. *D3.1 – Available methods, tools and mechanisms*. Technical report, CoeGSS, 2016.
- [3] S. Wolf, D. Paolotti, T. Michele, M. Edwards, S. Fürst, A. Geiges, A. Ireland, F. Schütze, and G. Steudle. *D4.1 – 1st report on pilot requirements*. Technical report, CoeGSS, 2016.
- [4] M. Edwards, S. Fürst, A. Geiges, L. Rossi, M. Tizzoni, and E. Ubaldi. *D4.2 – 2nd report on pilot requirements*. Technical report, CoeGSS, TBD.
- [5] S. Fürst. *Comparing Repast HPC/Pandora*. Technical report, CoeGSS, 2016.
- [6] R. Schneider, S. Gogolenko, M. Gienger, and B. Koller. *CoeGSS – ABM Software stack*. Technical report, CoeGSS, 2016.
- [7] X. Rubio-Campillo. *Pandora: A Versatile Agent-Based Modelling Platform for Social Simulation*. International Conference on Advances in System Simulation – SIMUL, 2014, pp.29-34. ISBN: 978-1-61208-371-1
- [8] J.T. Murphy. *High Performance Agent-Based Modeling in Repast HPC*. Seminar, Argonne, 2014. URL <https://www.ci.uchicago.edu/events/high-performance-agent-based-modeling-repast-hpc>.
- [9] *Grant Agreement No. 676547: CoeGSS*. Report, European Commission, 2015
- [10] M. Gienger, N. Meyer, S. Petruczynik, R. Januszewski, A. Cheptsov, B. Koller. *D5.1 – Definition of the CoeGSS Operation Environment*. Technical report, CoeGSS, 2015
- [11] M. Gienger, B. Karaboga, P. Wolniewicz. *D5.2 – First Operation Report*. Technical report, CoeGSS, 2016.
- [12] F. J. Nieto, B. Karaboga, M. Gienger, A. Rivetti. *D5.9 – Initial Portal Design*. Technical Report, CoeGSS, 2016.
- [13] F. J. Nieto, B. Karaboga, M. Gienger, P. Wolniewicz. *D5.10 – First Portal Release*. Technical report, CoeGSS, 2016.

List of tables

Table 1 – Pre- and post-processing servers on Hazelhen@HLRS..... 28

List of figures

Figure 1 – Web-interface of the portal’s frontend..... 10
 Figure 2 – Submission form: 1st stage – providing general information..... 19
 Figure 3 – Submission form: 2nd stage – providing a localization information 20
 Figure 4 – Data presented in a map form 20
 Figure 5 – COVISE Map Editor and Renderer 23
 Figure 6 – COVISE/OpenCOVER Integration..... 25
 Figure 7 – ReadPandora sample dataset..... 26
 Figure 8 – Showcase of PPT link from within a CAVE..... 27