CoeGSS

Centre of excellence

D3.2: First specification of new methods, tools and mechanisms proposed for the support of the application user and programmer

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/`
Project start date	2015-10-01
Duration	36 months
Due date	2016-03-31
Dissemination level	Public
Nature	Report
Version	1.5
Work Package	WP3
Leading Partner	Chalmers (Patrik Jansson)
Authors	Patrik Jansson (editor), Marcin Lawenda, Eva Richter, Uwe Woessner, Cezar Ionescu, Michał Pałka, Ralf Schneider, Devdatt Dubhashi, Michele Tizzoni, Daniela Paolotti, Steffen Fürst, Margaret Edwards
Internal Reviewers	Pawel Woniewicz, Michael Gienger, Giulio Cimini
Keywords	HPC, Big Data, Domain Specific Language, Synthetic Information System, Scalability, Visualisation, Co-design
Total # of pages:	54

Version History

	Name	Partner	Date
From	Patrik Jansson	Chalmers	
Version 0.9	1st internal review		2016-03-11
Reviewed by	Pawel Woniewicz	PSNC	2016-03-15
	Michael Gienger	HLRS	2016-03-15
	Giulio Cimini	IMT	2016-03-16
Version 1.0	2nd internal review		2016-03-24
Reviewed by	Pawel Woniewicz	PSNC	2016-03-30
	Michael Gienger	HLRS	2016-03-28
	Giulio Cimini	IMT	2016-03-29
Version 1.5	Ready to upload		2016-03-30
Approved by	ECM	UP	2016-03-30

Abstract

Work package 3 is a research and development work package with an overall aim to provide a set of tools and methods supporting the work of application programmers and GSS end users. This report is a living document, and the release at project month 6 is the second deliverable (D3.2) of work package 3 (WP3).

The first WP3 deliverable (D3.1) was about the state-of-the-art: methods, tools and mechanisms (MTMs) available off-the-shelf at the start of the CoeGSS project. With this report (D3.2) we propose new MTMs based on the “gap” between WP3 (tasks T3.1–T3.6) and WP4 (the pilots).

We start with a description of the CoeGSS workflow and then proceed through the six tasks of WP3 in the same order as in D3.1. For each WP3 task we first describe the requirements from the pilots which are relevant for that task, then the gaps to fill and finally the proposed solutions. Then, for completeness, we include the three pilots’ views of what the tasks of WP3 could contribute.

Table of Contents

Table of Abbreviations

ABM	Agent-Based Model
ACM	Awareness Creation Module
CKAN	Comprehensive Knowledge Archive Network (a data management system)
CoSMo	Complex Systems Modelling (a site in CoeGSS)
CoeGSS	Centre of excellence for Global System Science
D3.1	CoeGSS Deliverable 3.1 on Available MTMs
D4.1	CoeGSS Deliverable 4.1 on Pilot Requirements
DoW	Description of Work
DSL	Domain Specific Language
GIS	Geographic Information System
HDFS	Hadoop Distributed File System
HDF5	Hierarchical Data Format 5 (a smart data container)
HLRS	High-Performance Computing Centre Stuttgart (a site in CoeGSS)
HPC	High Performance Computing
HPDA	High Performance Data Analysis
KBM	Knowledge Base Module
MTMs	methods, tools and mechanisms
SIS	Synthetic Information System
VNC	Virtual Network Computing (a graphical desktop sharing system)
VR	Virtual Reality
WP	Work Package

1 Introduction

WP3 is a research and development work package supporting, directly and indirectly, the work of application programmers and GSS end users. The overall objectives of this work package for the full three year period are the following according to the DoW (slightly edited):

- To propose a prototype version of a heterogeneous environment consisting of HPC infrastructure and cloud storage to be used for scientific use cases (Chapters 2, 3)
- To provide enhanced fault tolerance skills in the proposed architecture (Chapter 3)
- To keep appropriate scalability for future large applications demanding a big data approach by increasing data efficiency (Chapter 4)
- To develop data layer tools and services with a unified interface to the underlying technologies (Chapter 4).
- To provide remote and immersive visualisation (Chapter 5)
- To provide DSLs for assembling GSS simulations (Chapter 6)
- To develop validated numerical methods for GSS simulations (Chapter 7)
- To develop a clear concept and support services for the hardware / software co-design of future needs coming from the users' communities (Chapter 8)

This report is a living document and the release at project month 6 is the second deliverable (D3.2) of work package 3 (WP3). The second release in month 21 will be D3.3 and the third release in month 31 will be D3.4. The first deliverable (D3.1) was about the state-of-the-art: methods, tools and mechanisms (MTMs) available off-the-shelf at the start of the CoeGSS project. With D3.2 we propose new MTMs based on the "gap" between WP3 (research tasks T3.1–T3.6) and WP4 (the pilots).

The pilots work package (WP4) also has a living document whose first release was reported as D4.1 (already in project month 3). That initial report identified requirements of the pilots on CoeGSS as a whole and in this report (D3.2) the three pilots expand on these requirements in Chapters 9 to 11.

In CoeGSS the High Performance Computing community (here represented by WP3) meets with the Global Systems Science community (represented by WP4). The intention is for D3.2 to be a first step towards bridging the gap between the two communities. We start with a description of the common CoeGSS workflow (in Chapter 2) and then proceed through the six tasks of WP3 in the same order as in D3.1 followed by the three pilots in the same order as in D4.1.

Executive summary

In Chapter 2 we present a common "CoeGSS workflow" which for a particular use case (like our pilot studies) entails the following steps: collect, clean and store input data, use a Domain Specific Language (DSL) to specify a suitable Synthetic Information System (SIS), implement (create) the SIS, run the simulation, analyse the resulting data, and finally visualise the results. The process is controlled by a Graphical User Interface and an Application Programming Interface.

Each of the six following chapters describe requirements, gaps and proposed solutions from the point of view of the six tasks of WP3.

- Chapter 3 presents methods and tools which support reliability and scalability of CoeGSS data management. Solutions are based on CKAN for data management, and the Hadoop Data File System combined with the parallel file system Lustre.
- Chapter 4 deals with data analytics for data collection, generation of synthetic populations and analysis of the resulting datasets. Methods include machine learning, discrete optimization, statistical significance testing and automatic parameter optimization.
- Chapter 5 deals with visualisation systems — both remote and immersive. The methods and tools are collected in the CoeGSS Visualisation Toolbox connecting to the CoeGSS portal, GLEAMviz, the R-project and using the Apache Hadoop software library and the Apache Cassandra database.
- Chapter 6 describes how DSLs can be used to specify Synthetic Information Systems, to translate between the formats required by different tools and to test, verify and validate the results. Methods and tools include interpreters, grammars, property-based testing, and model based testing.
- The focus of Chapter 7 is on ensuring validity and correctness of CoeGSS methods and tools. Methods include interval analysis, model based testing, property based testing, high assurance software through types and monadic dynamical systems.
- Chapter 8 describes how CoeGSS will work together with the users to design and to some degree also implement hardware, software and analysis solutions to meet their needs. The methods and tools will be collected in a knowledge base, tutorials and courses.

Chapters 9 to 11 contain short descriptions of what the three pilots of WP4 see as possible contributions from the WP3 tasks.

- The Health Habits pilot (Chapter 9) needs efficient management of the output data stream, integrated computation of confidence intervals and visualisation of a large parameter phase space.
- The Green Growth pilot (Chapter 10) has started working with HLRS on co-design of a simple prototype model using a synthetic population of cars. Geographic visualisation on a global grid is needed with navigation also in time and parameter space to compare multiple runs.
- Finally, the Global Urbanisation pilot (Chapter 11) starts from the well-established CoSMo modelling and simulation platform (implemented in C++). From WP3 CoSMo is primarily interested in data analytics, massively parallel parameter scans, sensitivity analyses, disaggregation and interactive visualisation.

This deliverable is the first release of a living document which will evolve during the three project years. From what is now a collection of six rather distinct R&D tasks and three pilot study tasks we expect to see an increasingly coherent picture emerge from the Centre of excellence for Global System Science. As we learn more about the needs of the pilots and as we develop new methods and tools for Global Systems Science on HPC we will also be able to cover new areas and find innovative ways of generating value for new partners and customers. At the same time we will push the fields of High Performance Computing and High Performance Data Analytics towards handling new workloads and requirements and towards new application domains. Interesting times lie ahead!

2 Workflow

2.1 Language and terminology

At the start of the project we recognized that participants coming from different areas are using different jargons or scientific dialects. For example, “workflow” means something different for GSS and HPC people. (See also the Table of Abbreviations on page 5.) We have actively worked on unifying the meanings where possible, and otherwise at least creating awareness of the different meanings.

For HPC people the processing stages of the workflow are perceived as black boxes with input and output. The important part is to assure a proper data flow including input-output data compatibility: the output file from the preceding processing block should serve (optionally after treatment) as the input file to the succeeding block. HPC people do not investigate too deeply how data is processed inside the application.

GSS people are mostly concentrated on data analysis inside the particular application (or other steps which must be conducted to achieve a specific level of data treatment) while aspects related to the input-output data compatibility or storage place have secondary importance. In the GSS world, **scenarios** would be seen as different possible futures of the world (or a global system). This term is e.g. used frequently in the climate change context. There, it is often preferred not to attach probabilities to these scenarios, but the word scenario itself suggests that there is more than one possibility, and that we don’t know which one will come true. In the GSS world, the **synthetic population** is the set of agents, created from data, then fed into an Agent-Based Model (ABM) to run simulations, and this whole thing together is the SIS.

Concluding, it can be stated that there are two different approaches:

- data flow - characteristic for HPC people - where proceeding sequence in terms of data compatibility is important
- goal oriented - characteristic for GSS people - where implemented methods and achieved results are the most important.

2.2 Introduction

The Global Systems Science scenarios consist of many processing steps that must be inherently performed in order to get correct results. The identification of steps and their factual order is essential to design the system which properly succours the entire process. The analysis was started with a general HPC workflow where vital GSS processing phases are identified. Then, each pilot (use case) was decomposed into processing phases and presented on the graph including conditional repetition of scenarios which require recalculation. Finally, all workflows are collated together in order to determine the common features and possible unification. As a consequence, the main workflow objectives are:

- Organisation of the data processing and analysing scheme
- Facilitation of data management within one scenario (workflow)

Initially, it was assumed that all our use cases share a similar control flow where we can distinguish analogous operations. These operations can be connected in the sequence of events creating a predefined processing scheme. Based on this characterisation we develop a framework to generate customised synthetic populations for GSS applications. The workflow depicted in Figure 2.1 presents a general (high) level of abstraction which works as a starting point for further discussion and to create a common view of the problem.

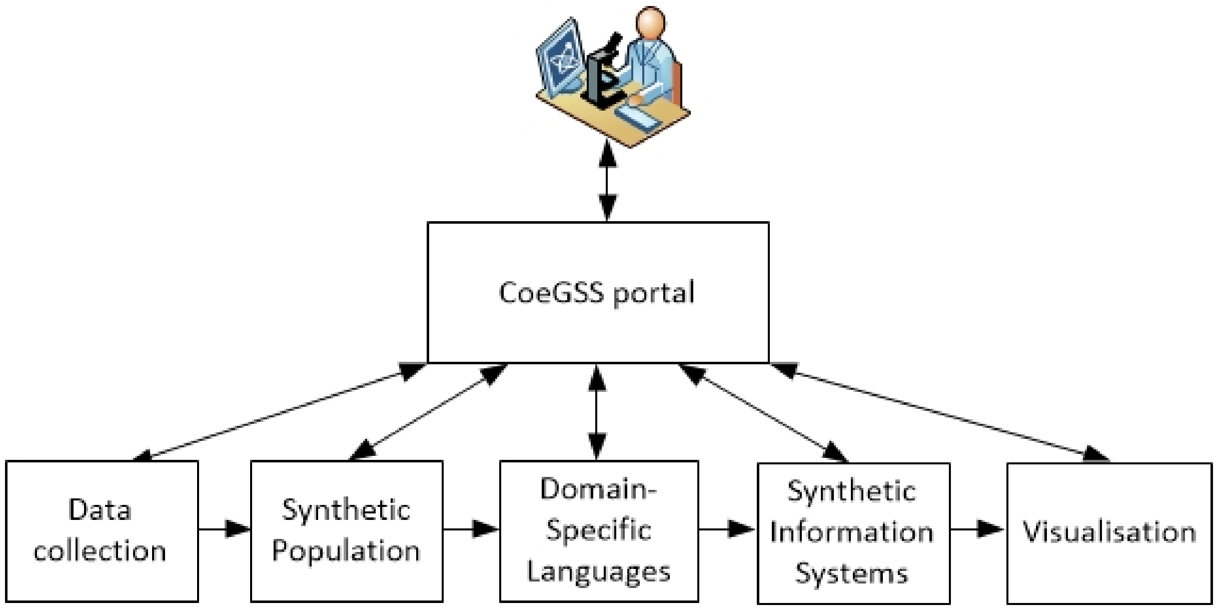

Figure 2.1: CoeGSS general workflow

The entire process is started with procuring input data of different kinds and from different origins. Currently, data are collected manually by browsing different sources and putting them together. Next, data and metadata must be stored in a data management system, which becomes a source of input for the processing systems. Then data are aggregated and described by metadata which explain and complement their meaning (e.g. date stamp, place of origin, number of people took a part, format info). CoeGSS will support this process via the portal by providing a data pool, and different tools which facilitate data storage, re-formatting and tagging with metadata. Meanings and relations between the data (including static and dynamic aspects of simulations) are described by a Domain Specific Language (DSL), which specifies a synthetic population and action patterns for the modelling systems. The DSL will allow users to describe the agents and the simulation at a higher level of abstraction, and then the DSL compiler would make the necessary changes to the low-level processing framework, in much the same way as a compiler of a high-level language can generate code written in C.

When data are collected and the agents are specified by the DSLs the procedure is ready to run a simulation of a Synthetic Information System (SIS). The process is performed on HPC systems, usually in batch mode. The HPC system is coupled with a data management system in order to efficiently provide input data, a location to store processing results, and to support processing relocation in case of an emergency. The SIS output data is sent to the CoeGSS visualisation toolbox (Chapter 5) or to statistical systems in order to perform a further analysis of the achieved results (Chapter 4). Visualisation software is also located on HPC resources and it is run in interactive mode. The graphical output is sent from the HPC systems to end users via the portal using different solutions for remote control and access to graphical desktop applications like RealVNC, TightVNC, UltraVNC.

The portal serves as a user interface (an access point) for controlling and monitoring the whole workflow. Access to services is offered in the marketplace where three predefined use cases (Health Habits, Green Growth and Global Urbanisation) are placed. Outside of the mentioned services, a set of tools is also available. They are used e.g. for preparation of input data or re-running, out of the box, visualisation without having to restart the whole scenario.

2.3 Batch mode versus interactive mode

Operations represented by the blocks in Figure 2.1 can work either in batch mode or in interactive mode. Most activities (like data preparation, adaptation to DSL and visualisation) will be made interactively, while others (e.g. SIS) will be executed in batch mode (see Figure 2.2). Some of them (e.g. data preparation, adaptation to DSL) can be executed on regular remote (virtual) servers but computation related to SISs and advanced visualisation need HPC efficiency.

Batch mode

In batch mode a task is submitted to the HPC queuing system, waits for resource allocation and is executed without human interaction. An end user is just waiting for the result which is communicated either in monitoring systems or by mail messages.

Interactive mode

Interactive processing is closely related to a researcher's (virtual) presence. All work is done by the scholar with support of specified tools. Tools are run on remote machines by the workflow system. If any scheduling process is considered in the background of the workflow, it must take into account the researcher's availability. For example, if a scientist wants to analyse data in a visualisation system (a significantly resource demanding process), this activity must be scheduled according to the work time and assumed duration time.

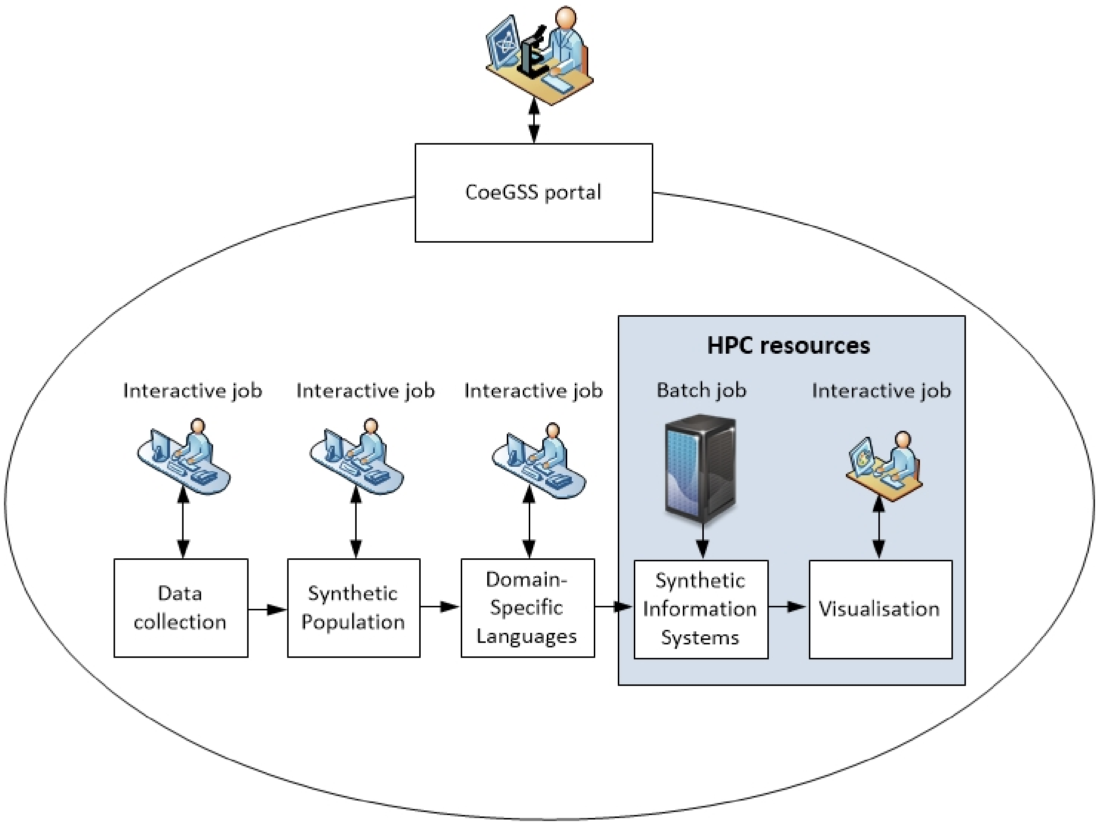

Figure 2.2: Batch vs interactive mode in GSS scenario

2.4 Data management within the workflow

Each of the processes in the workflow produces data (output files), which should be stored in an orderly manner in the data management system (see Figure 2.3). This allows to reuse the data when necessary.

Data are organized into groups defined by workflows (e.g. workflow id) and assigned to a specific part of the scenario.

All data are described by metadata. An example structure is as follows:

- workflow name, workflow id
- processing block name, id: data, DSL, SIS, Visualisation
- input/output data
- date + time of processing / preparing
- status: under preparation, ready to process, partially processed

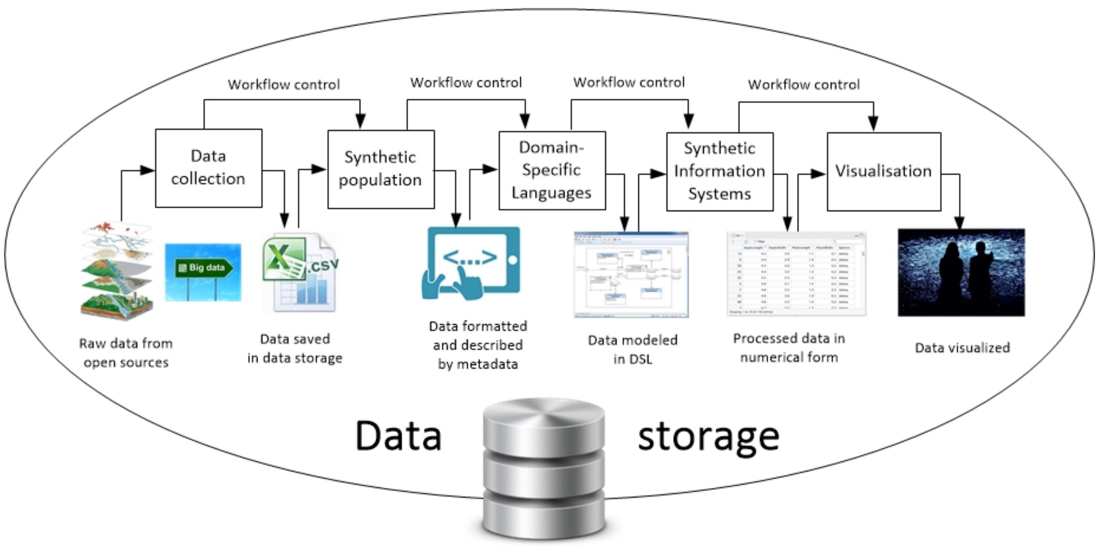
Workflow control
Workflow control
Workflow control
Workflow control
Data collection
Synthetic population
Domain-Specific Languages
Synthetic Information Systems
Visualisation
Big data
.CSV
Raw data from open sources
Data saved in data storage
Data formatted and described by metadata
Data modeled in DSL
Processed data in numerical form
Data visualized
Data
storage

Figure 2.3: Data flow in GSS workflow

2.5 Use case workflows

From the global systems point of view, the GSS workflow should include further elements and many more links in that it is iterative rather than "linear": The first element is a global system of interest. It will come with large overarching research questions, like, in our case "How is Green Growth possible, i.e. how can emissions be seriously reduced but at the same time poverty be reduced as well, economic and social development furthered rather than hindered?" The second element is a conceptual model. A researcher decides which parts of the system to look at, which to leave out, and plans how to represent the parts which must be represented. The analyzed part of the system comes with a more concrete research question, like in our case "how will the global car population and the emissions from it evolve, and how might policy interventions influence this evolution?". In specific situations the analysis process can be started directly from the more concrete question, e.g. because a study with such a question is commissioned by some stakeholders, but then the researcher would only be the one they ask because he has experience relating to the larger overarching question. Having specified a rough model, one looks for data, and depending on what one finds, the model may need to be refined or even completely changed. This is shown by a first arrow backwards from data to conceptual model in Figure 2.6 explained in the Green Growth subsection (2.5.2).

2.5.1 Global Urbanization workflow

Global Urbanisation pilot study into the systemic impact of infrastructure decisions on key urban performance indicators such as congestion, real estate prices and emissions. See Figures 2.4 and 2.5.

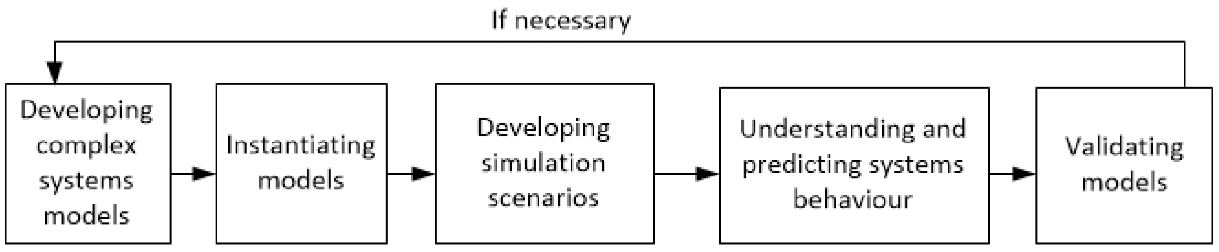
If necessary
Developing complex systems models
Instantiating models
Developing simulation scenarios
Understanding and predicting systems behaviour
Validating models

Figure 2.4: Global Urbanization workflow

Tool: CoSMo application

Input data: publicly available data

The most important steps in the workflow

Step 1. Developing complex systems models

- Define categories of entities, behaviours, hierarchies, interactions and scales
- Generate engine

Step 2. Instantiating models

- Integrate raw data from external sources (files or database)
- User defined data

Step 3. Developing simulation scenarios

- Define parameter exploration
- Define model perturbation
- Implement model observers

Step 4. Understanding and predicting systems behaviour

- Conduce numerical experimentation through sets of simulations
- Analyse results

Step 5. Validating models

- Verification and validation
- Regression testing

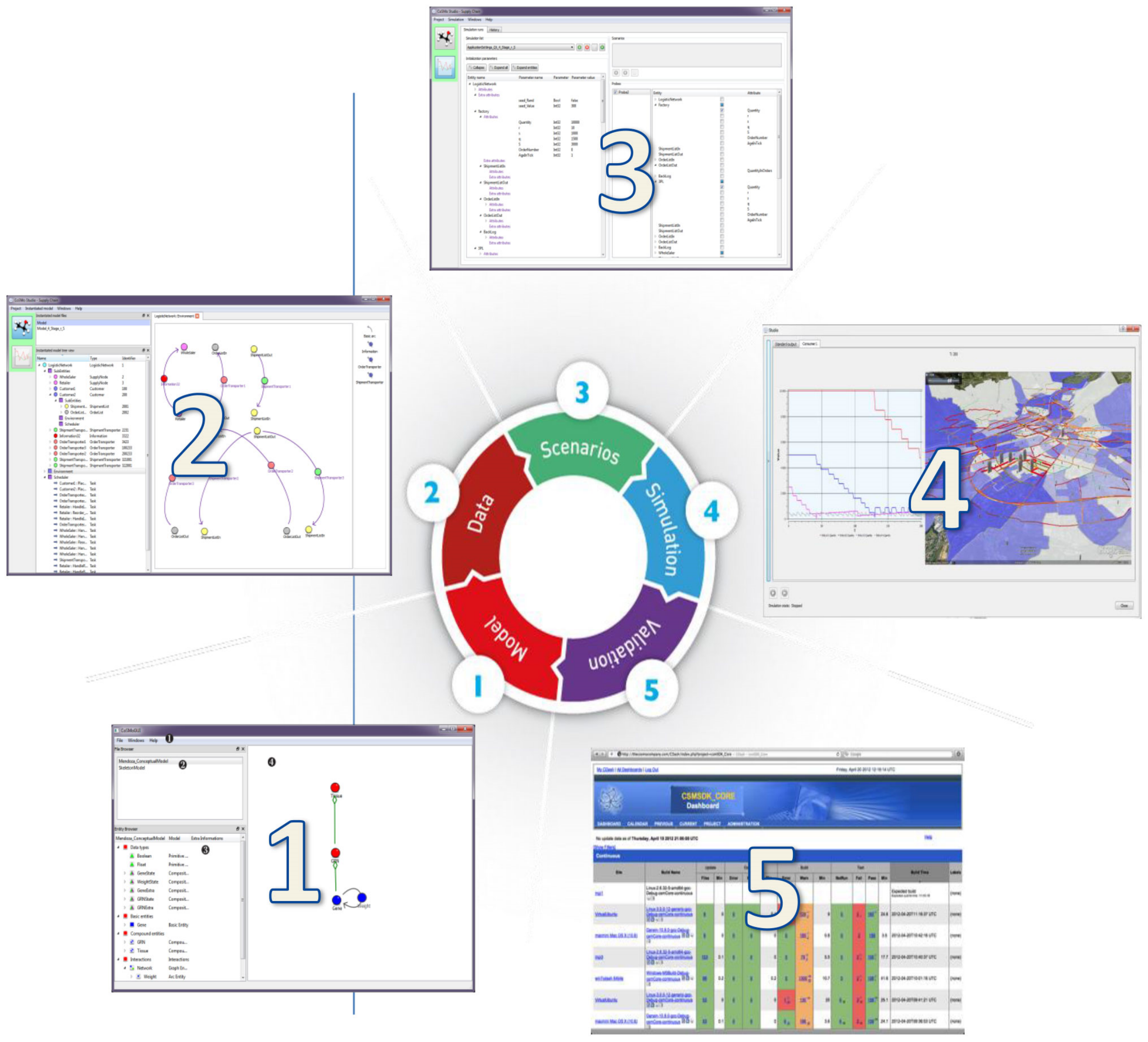

Figure 2.5: CoSMo: A unique modelling and simulation process.

In order to allow for more freedom, model observers are defined with the simulation but usually not in the first step. For instance after having defined the conceptual model of a city and instantiated it over London, simulations can lead to counter-intuitive results requiring to add more model observers - here it is precious not to need re-initiating the whole process.

2.5.2 Green Growth workflow

A pilot study into the possibility of green growth, i.e. increased well-being in the economic, ecologic and social dimension, investigating the diffusion of initiatives such as feed-in tariffs, green business strategies, and individual lifestyle changes. There is a nested workflow working according to the rule — build a simple SIS, try it out, make it more complex, try it out again, etc. The "inside" workflow is more similar to the one presented in Figure 2.1, with the addition of a "conceptual model" in the beginning and some iterations in between. It is pretty similar also to the workflow of Health Habits (T4.3) as described in Section 2.5.3.

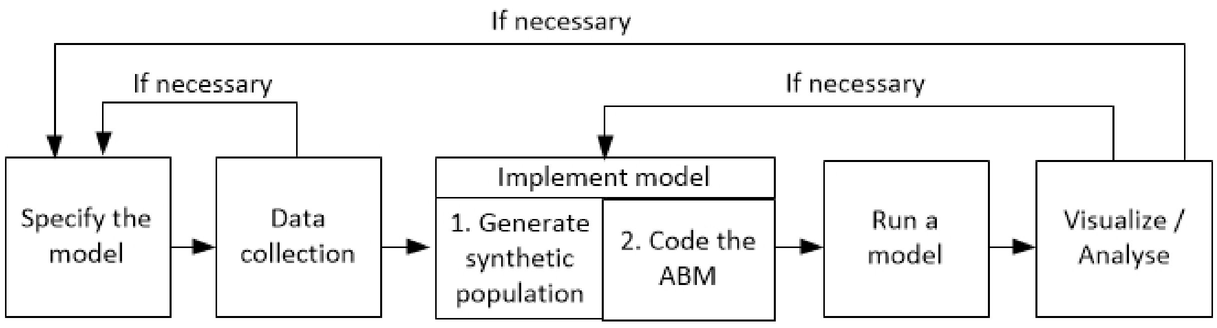

Figure 2.6: Green Growth workflow

Tool: Pandora

Input data: publicly available data (e.g. data.worldbank.org, oica.net, openstreemaps.org)

The most important steps in the workflow

Step 1: Specify the model (agents, characteristics of agents, groupings of agents, interactions, environment, etc.).

Step 2: Data (collect and pre-process the data to instantiate the model)

Step 2a: if necessary, go back to Step 1 to make changes

Step 3: Implement the model

Step 3.1: generate a synthetic population

Step 3.2: code the ABM (using Pandora)

Step 4: run the model (possibly for sets of parameters, e.g. for sensitivity analysis)

Step 5: visualise/analyse model run output (we initially use Cassandra, but work closely together with the visualisation task to suit the individual needs)

Step 5a: if necessary, go back to Step 3, or even Step 1 and make changes

Building a new SIS, one can expect several up to very many iterations as mentioned in Step 5a. The DSL, once available, will be used in Step 1 and Step 4.

Each specific model (each SIS) needs generation of a synthetic population, an agent-based model and running the SIS with different sets of parameters. Many runs are required to explore what "an ABM does" (not in the sense of how it performs, but in the sense of what its output looks like). It can be seen as a massive iteration step — see what happens in simulations, go back to the model itself to make changes and recalculate it again.

2.5.3 Health Habits workflow

The main goal of this pilot is to create a Synthetic Information System, integrating large and heterogeneous data sources, that will allow to describe and model the spread of relevant health habits (e.g. smoking, obesity) at the population level in Europe. (See Figure 2.7.)

Tool: Design and implementation by ISI. The GLEAM/GLEAMviz[1] model will not be used to simulate the spread of smoking or obesity and, more in general, will not be integrated into the Health Habits pilot. GLEAM will only be used as a reference tool, or to share some relevant data set such the population distribution.

Input data: not completely defined yet. Several demographic (age, gender) socio economic (income) and health indicators of European population. Among others, a high-resolution population database - the Gridded Population of the World project by the Socio-Economic Data and Applications Center (SEDAC)[2].

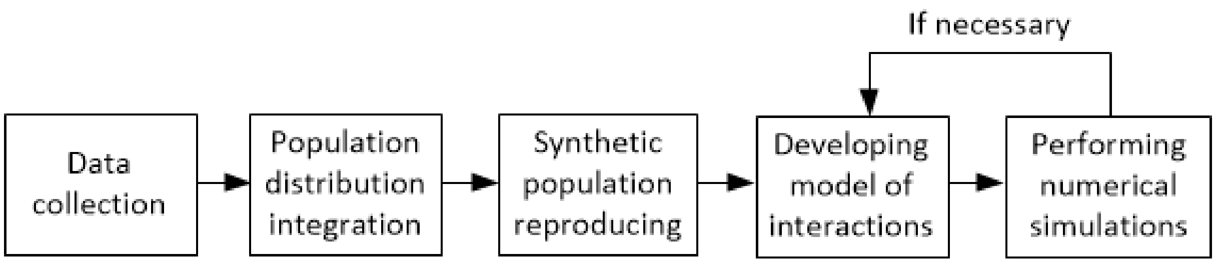
If necessary
Data collection
Population distribution integration
Synthetic population reproducing
Developing model of interactions
Performing numerical simulations

Figure 2.7: Health Habits workflow

The most important steps in the workflow

Step 1: Acquiring data from a variety of statistical sources.

Step 2: Integration of population distribution with additional information on relevant health statistics and creation of the synthetic population.

Step 3: The synthetic population will reproduce the prevalence of smoking or obesity in the country of interest, by age groups and by region.

Step 4: Assigning agents to relevant social clusters.

Step 5: Developing model of interactions between agents.

Step 6: Performing numerical simulations with the model. If necessary go step 5, modify model and perform simulation again.

[1]GLEAMviz simulator: `http://www.gleamviz.org/`.
[2]`http://sedac.ciesin.columbia.edu/data/collection/gpw-v3`

2.6 Integration of concepts

As a final stage of the current analysis an collation of the different workflows has been performed. A graphical juxtaposition is presented in Figure 2.8.

The presented outcome so far is based on information coming from an analysis of the pilot workflows. No modifications were introduced in the original process flow and division into stages. The pilots use different names for particular processing stages but many similarities can be observed. Fundamentally, one processing block in the general CoeGSS workflow corresponds to one or two blocks in each pilot workflow. The integration of the pilot workflows needs further study. This will be done in the direction of workflow unification, automation and reuse. It is expected that this work should facilitate future manual and automatic processing under CoeGSS system.

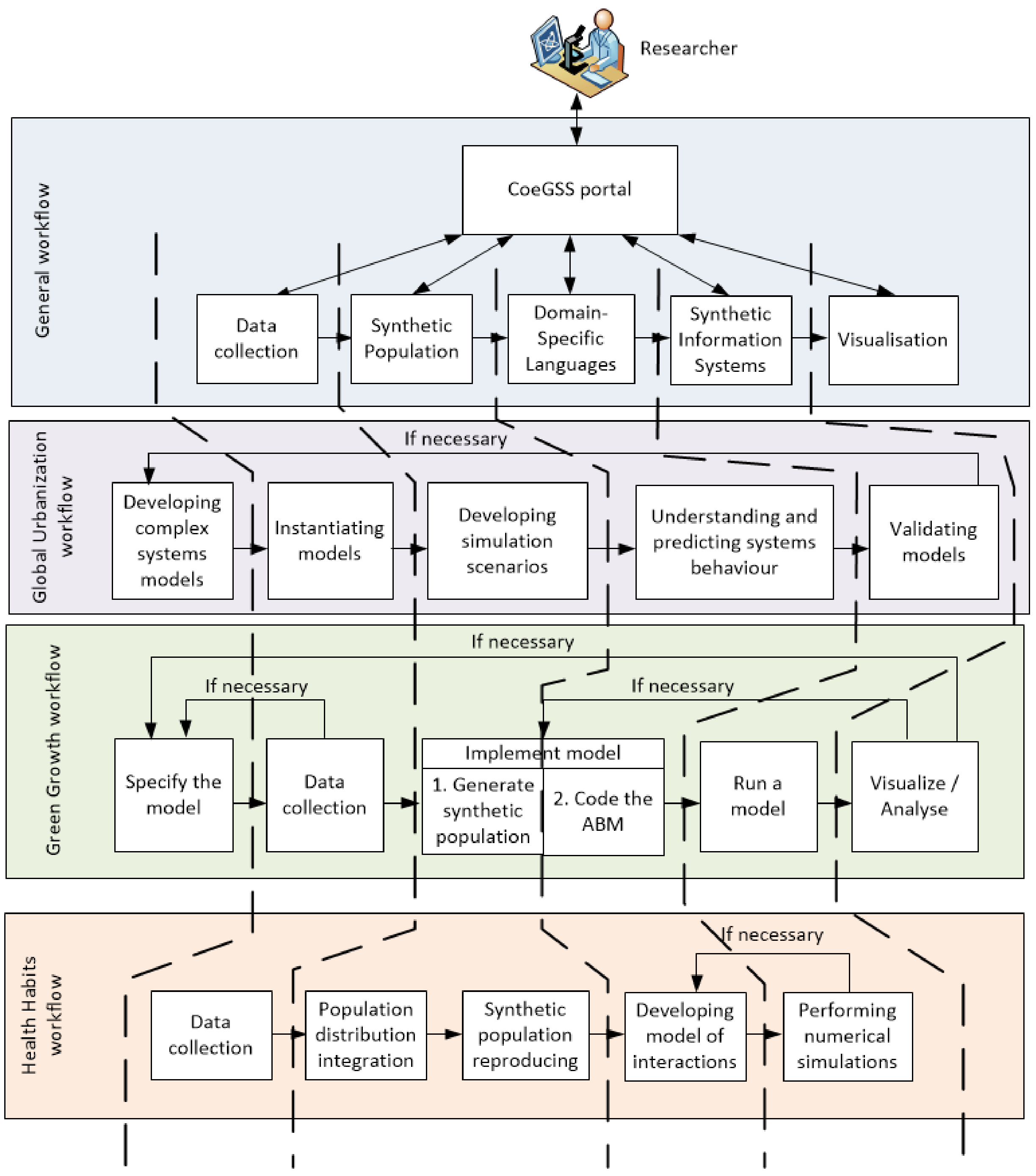
Researcher
General workflow
CoeGSS portal
Data collection
Synthetic Population
Domain-Specific Languages
Synthetic Information Systems
Visualisation
If necessary
Global Urbanization workflow
Developing complex systems models
Instantiating models
Developing simulation scenarios
Understanding and predicting systems behaviour
Validating models
If necessary
Green Growth workflow
If necessary
If necessary
Specify the model
Data collection
Implement model
1. Generate synthetic population
2. Code the ABM
Run a model
Visualize / Analyse
Health Habits workflow
If necessary
Data collection
Population distribution integration
Synthetic population reproducing
Developing model of interactions
Performing numerical simulations

Figure 2.8: Conceptual integration of workflows

3 Enhanced Reliability and Scalability

3.1 Introduction

Providing reliable and updated results to project customers or stakeholders is, not only in the GSS world, of the highest importance. In this chapter we focus on methods and tools to enable GSS applications reliable and scalable in terms of both: computing and data storage. Reliability and scalability can be analysed from different perspectives: computation and storage.

The computation perspective needs relevant software profiling (software tuning) and fault tolerance handling (by using an appropriate processing model or checkpointing). As it was mentioned in the previous chapter the GSS workflow consists of many processes. Each of them reads input data needed for processing and finally produces new output files in every iteration which must be stored in an orderly manner in data management systems. Moreover, it is assumed that GSS scenarios are usually performed many times in order to compare and verify results. It implicates additional space-related requirements.

The Data Management System (DMS) used for GSS pilots is required to provide sufficient space for storing large amount of data (up to petabytes in future applications) like initially collected information, intermediate files, and results in diverse formats (text, pictures, maps, videos). Next, the DMS must be able to serve very intensive computation (rapid read and write datasets) and, in case of failure, recover full functionality of the data management system. Collection of input data and its subsequent processing is a time-consuming process requiring a lot of expertise. Data protection in a case of system failure is a priority which is described in the following sections.

3.2 Requirements

Currently, in all use cases data are collected manually. Data are gathered from publicly available, sources including:

- Global Urbanisation: e.g. INSEE[3]
- Green Growth: e.g. World Bank[4], OICA [5], Open Street Maps[6]
- Health Habits: public sources e.g. Eurostat

Data are stored in several formats: csv, xml, hdf5, geotiff for geo-reference data.

At the moment the size of the input datasets is not significant. In the future, due to the system development and introduction of automation process, the amount of data will increase to giga- or terabytes. Processing of GSS scenarios leads to computation of huge datasets even when the input data is relatively small. Moreover, we can expect a tremendous growth of the data size as well as their heterogeneity — taking into account different data sources, e.g. automatically parsed data streams from social media. Automation plans and improved computational scalability is essential to support the amount of data which must be calculated as we increase the number of agents. It will allow efficiently use of available resources like CPUs, memory, network, storage and reduce calculation time. It drives us to two main problems which should be solved in the proposed GSS framework: efficient data management as well as reliable and scalable computation.

[3]http://www.insee.fr
[4]http://data.worldbank.org/
[5]http://www.oica.net/
[6]http://www.openstreetmap.org/

3.3 Gaps

As mentioned above, presently, input data for pilots are collected manually. Data are stored in files on local workstations. No data system providing robust, reliable and effective data management is used at the moment. The problem of the computation scalability is also uncovered in terms of CPU numbers (Green Growth and Global Urbanization use cases) or data access (ISI use case). Number of used CPUs are expanded as long as currently implemented software and ineffectively used resources (memory, storage) permit. Furthermore, taking into account already mentioned aspects of data computation and storing, it is essential to provide services for GSS processing that will handle these requirements.

3.4 Solutions

The suggested solution includes several systems which provide continuity of the operation in case of failure and effective environment for large-scale computation.

For the primary data storage system **CKAN** was selected. It offers extensive functionality, described below, very needed and convenient to use from the CoeGSS project point of view. **Hadoop** with **HDFS** and **Lustre** delivers a reliable and robust data storage system. An appropriate processing model including snapshots and a checkpointing system enables fast system operating after failure. Tools for software profiling assure that resources are effectively used.

3.4.1 Tools for efficient data management

CKAN is an open source data management tool providing many features which are suitable to application in GSS solutions. It facilitates data exploration in many ways and makes them presentable via websites or APIs. Provided functionality can be easily extended by new plugins developed by programmers.

CKAN features essential for GSS:

- integrated storage
- programming API
- library of extensions providing additional functionality
- web interface, integration with CMS
- visualisation with tables graphs and maps
- analysis: statistics, usage metrics
- data access control
- importing data from remote sources (e.g. web sites)
- searching by keyword or tag
- searching geospatial data
- data versioning
- storing raw data and metadata
- interaction with other CKAN nodes

- community building by commenting and following datasets
- Open Source license

Data storage

Data could be stored either in an organised structure of file and directories or in the PostgreSQL database. It is possible to store data in structured and unstructured format.

Metadata

Data stored in the CKAN system are described by a set of metadata. There are standard fields like: identifier, title, group, description, data preview, history revision, tags, API key, license. There is also a possibility, which will be used in CoeGSS, to define extra fields to hold additional information (e.g. geospatial features).

API

A RESTful JSON API allows developers to explore data on different ways:

- querying,
- information about dataset,
- retrieving,
- using data without downloading etc.

The API is accessible for authorised users only which increases the security of data access. The API is also well documented and available via the website CKAN.org.

Web interface

The web interface is a very important feature from the point of view of user interoperability (WP5). It will allow data storage integration with the GSS portal and hand over any vital functionality. Data can be visualised in many forms: as tables, graphs, maps, which significantly facilitates the analysis process.

Data access

CKAN provides fine-grained access control. Datasets can be defined as public or private. Private data are visible only to authorised users e.g. logged to the website.

3.4.2 Tuning procedure

The CKAN system in the standard configuration works comfortably with smaller groups of datasets. In order to increase scalability, and e.g. import millions of datasets efficiently, performance tuning is needed. Tuning is made on different levels: configuration files, database table design, and database configuration. When tuning process is complete CKAN works very effectively. The tuning is required for effective storage usage and needs to be tailored to CoeGSS needs. The tuning process is described in details on the CKAN wiki[7].

[7] https://github.com/ckan/ckan/wiki/Performance-tips-for-large-imports

3.4.3 Scalability

Data scalability

The proposed solution is based on a combination of Hadoop (HDFS) and Lustre in order to combine the advantages of both systems. In the TeraSort test[8] this combination gives 30% speedup (50% when transfer time is considered) when compared to a solution without HDFS. This test measures the amount of time to sort one terabyte of randomly distributed data.

Advantages of Hadoop:

- huge volumes of data
- huge number of files
- data analysis — MapReduce
- HDFS — Hadoop Data File System
- serves for structured and unstructured data

Hadoop allows processing of large data sets via processing nodes. A disadvantage which need to be taken into account in further system design is its weakness in ad hoc queries. Hadoop has a flexible file system and the very capable and flexible MapReduce framework for processing large data sets. The biggest known production scalability for the Hadoop File System (HDFS) is up to 200PB of storage and a cluster of 4500 servers with billion files[9]. HDFS and MapReduce constitute the strength of Hadoop. Data are distributed on several storage servers what allows parallel processing instead of sequential. MapReduce provides processing software directly on the storage nodes.

There are two stages of the processing:

1. Mapping an operation of the distributed parts of data
2. Reducing (aggregating the results) and returning an answer back

There are many associated projects with Hadoop which facilitate its usage like:

- **Hive** (https://hive.apache.org/) allows managing and querying data stored in distributed storage
- **Pig** (https://pig.apache.org/) provides a high-level language for data analysis
- **HBase** (https://hbase.apache.org/) database for distributed data store to allow hosting very large (billions of rows) tables.

[8]http://cdn.opensfs.org/wp-content/uploads/2015/04/Understanding-Hadoop-on-Lustre-Performance_Skory.pdf
[9]http://hortonworks.com/hadoop/hdfs/

Hadoop on Lustre Lustre is a parallel file system that facilitates HPC simulation environments by providing computer clusters with efficient storage and very fast access to large data sets. It uses object based disks for storage and metadata servers for storing file system metadata. Lustre allows to distribute very big files across many cluster nodes and uses a global name space. It is distributed under an open-source license. Lustre and Hadoop together permit to solve problems related with big data, harnessing the HPC power on very fast storage. However, there are also some disadvantages. The first of them is the overhead generated by HTTP calls to acquire data access by the Hadoop File System. The second disadvantage is the requirement for each Hadoop node to ensure a large local storage. One of the solutions for this problem is using the Intel Enterprise Edition for Lustre which provides a special adapter to overcome those drawbacks by implementing Lustre direct access within computations of MapReduce.

Processing scalability

The most commonly used method to increase software scalability is software profiling. There are many free and proprietary software profilers which enable analysis of many aspects of software activity e.g. CPU, I/O, MPI usage. One of the proprietary profilers (available at PSNC) is **Intel® Parallel Studio** (https://software.intel.com/en-us/intel-parallel-studio-xe) and it contains a set of tools and libraries for compiling and profiling of parallel applications. An example of an open source profiler is **aprof** (https://github.com/ercoppa/aprof) which allows to analyse and understand applications which scale as a function of the input data size. The **Vampir** (https://www.vampir.eu/) performance analysis framework is used to optimise applications including distributed ones. It provides a convenient graphical interface which facilitates the analysis process by providing information in charts which makes it easy to trace which part of an application takes most time.

3.4.4 Enhanced reliability

Processing reliability — fault tolerance.

GSS systems perform complex and lengthy calculations where the risk of crash of the computational system is significant. Fault tolerance (FT) systems prevent from interfering with operations in case of hardware and operating system failures. This approach is primarily intended for high-availability or life-critical systems. HPC systems offering fault tolerance functionality allows to prevent partial results and restart a computation later (from a previously saved snapshot) saving already consumed energy and time. Fault tolerance is also important from the point of view of energy consumption . Today's HPC systems consume huge amounts of energy which is wasted in case of a system fail. This waste can be avoided with an effective fault tolerance system covering data and computing levels.

Data reliability

Hadoop based on HDFS provides a reliable solution for storing a very significant amount of data (petabytes). The HDFS system works with three types of nodes: *name node*, *secondary name node* and *data node*. Name nodes are responsible for file system metadata management. The persistent state and checkpointing of the name nodes is the task of the secondary name nodes. The data nodes store data files in a series of 64MB blocks. Hadoop is able to detect different types of failure: hardware, software and human (e.g. unwanted operations like accidental deletion). There are also different methods to detect a failure like connection timeouts, checksums during reading from a disk or transmission. One of the options to recover data is using information saved in replicated blocks, which can be configured by the system administrators. The replication level can be configured by the administrator. Usually it is sufficient to use two replicas, for critical data this number can be increased to three or even four.

Processing reliability

Processing model Synthetic Information Systems used in the CoeGSS project are modelled by project participants themselves. This give us some degrees of freedom to implement a reliability mechanism by processing small chunks of data and by frequently saving the current processing state. Depending on the computational model the current status of the computation (output data) will be saved on the disk with a reasonable frequency. This frequency should be adjusted on one hand to the length of the entire scenario processing, and on the other hand to the length of processing of some part or parts of scenario. Data saved on the disk should allow to restart processing utilising a recently made snapshot.

External fault tolerance support The checkpointing allows to save the current status of the running task. This process is executed by the operation system. When a task fails, instead of initiating from beginning it is restarted from the recently checked pointed state. The process of checkpointing is carried out periodically i.e., checkpoints are kept and the process is executed from the recently saved state, once the system recovers from the fault. The Berkeley Lab Checkpoint/Restart (BLCR[10]) implementation combines user and kernel level checkpointing for a wide range of applications. A very interesting aspect of this solution is that BLCR does not require changes of the application code. The main focus is on parallel applications that communicate through MPI. The project covers the following areas: Checkpoint/Restart for Linux (CR), Checkpointable MPI Libraries, Resource Management Interface to Checkpoint/Restart, Development of Process Management Interfaces. BLCR is available under an open source license.

[10] `http://crd.lbl.gov/departments/computer-science/CLaSS/research/BLCR/`

4 Data Management / Data Analytics

4.1 Introduction

(Big) Data Analytics is a pervasive part of the pipeline of HPC for global systems science 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. Additionally, the storage of the data has to be taken into consideration, since the datasets will be so big that conventional relational databases will be unpractical. Since parallel computing will be needed due to the size and complexity of the calculations a good framework for distributed computing is needed.

There are also questions related to how to process the output data and how to deal with the uncertainties that originate in the input data and in modelling assumptions. In the following sections we explore these questions in more detail.

4.2 Requirements

All three pilots have to deal with very large populations and this makes parallelization, scalability and pre-processing of data very important. It is also necessary to be able to easily run the model with many sets of parameters to do parameter scans over a multi-dimensional parameter space.

Other important requirements are the ability to individually tailor the synthetic populations to the three different pilots, including aggregation of data from different data sources, and to decide which data analysis methods to apply to the results. There is also the aspect of how reliable the results will be: sensitivity analysis to find important parameters and to identify unimportant ones, and uncertainty analysis in order to establish confidence intervals for the outputs.

4.3 Gaps

From the requirements above the following gaps were identified:

- The data that will be collected from many different sources must be stored and organized in a consistent way, enabling the Centre's applications (including the pilots) to easily select and adapt it to their specific configurations.
- Since the quality and availability of the input data will vary, there will be gaps in the data that must be handled.
- Traditional relational databases are not suitable due to the size of the datasets. Instead, new Big Data management systems for unstructured data, such as MongoDB, Cassandra and Apache HBase, have to be used.
- Currently the pilots are not using any tools to quantify the uncertainty resulting from the varying quality of input data and the simplifications made in the models. But the stakeholders will need an indication of confidence intervals or scenario likelihood.
- The dynamic part of the SIS needs to be calibrated based on relatively sparse and aggregated data. This requires algorithms that deal with unstructured and incomplete data sets.

- Algorithms are needed to explore the massive multi-dimensional results of SIS simulations such as those performed by the pilots.

4.4 Solutions

It turns out that Apache Spark is able to fill most of our gaps. Spark is an open source cluster computing framework, based on the Hadoop ecosystem and originally developed at the University of California, Berkeley's AMPLab[11]. Spark has a strong support base both in academia and in industry. IBM has recently invested in data analytics specific to Spark[12] and Intel is supporting the work on optimising Spark for HPC[13]

Spark offers a fast and parallel distributed framework for data analytics that can be tightly integrated with an HPC cluster, although it has yet to prove that it scales as well as Hadoop. (We will closely follow the ongoing work within this field.) Additionally, to ensure that scaling is done in a data driven manner, we will use recent advances in nonparametric Bayesian methods [Hjort et al., 2010].

Spark also provides methods to aggregate data from different sources and works well with sparse data, reducing the problem with gaps in input data. Another advantage with Spark is that it supplies connectors for different data management systems, including MongoDB, Cassandra, HBase and also for relational databases.

The R package SparkR provides an interface between Spark and the popular statistical programming language R. R has a number of extensions that support data processing and machine learning tasks, in particular providing statistical significance testing and confidence interval analysis. These extensions will be used to quantify the reliability of the results.

Spark also includes MLlib, a machine learning library for large datasets that provides algorithms such as clustering, regression and Principal Component Analysis. MLlib will be used to perform basic machine learning operations such as clustering the output of the simulations and inferring underlying behaviours in the population.

In the coming months we will also evaluate the Bayesian analysis packages provided by Spark in order to estimate missing values, join various data sources and perform statistical inferences on the SIS output data.

[11] `https://amplab.cs.berkeley.edu/software/`
[12] `http://www.ibm.com/analytics/us/en/technology/spark/`
[13] `http://insidehpc.com/2015/11/berkeley-lab-to-optimize-spark-for-hpc/`.

5 Remote and Immersive Visualisation Systems

5.1 Introduction

This chapter gives an overview of relevant visualisation requirements from D4.1 and selected visualisation methods from D3.1. In general, the remote and immersive visualisation system will support partners and users within the CoeGSS project to explore, investigate and analyse GSS related data interactively with HPC support as required.

According to the CoeGSS general workflow (see Figure 2.1) the visualisation system is interfacing the simulation and data processing framework in two ways. First, the visualisation system is fed with data from the SIS, enabling users to investigate, browse and analyse achieved results interactively. In addition, the user interface will enable feedback to the simulation and data processing framework which, for example, then may have to rerun processing algorithms on remote HPC resources if necessary. As a result, the user is enabled not only to browse the data but to interactively process the data, to rearrange data sets or remap aggregations, for instance. Within the project the focus of task T3.3 is to provide remote and immersive visualisation to the consortium partners, to develop remote visualisation services in order to provide interactive access to HPC and visualisation resources, to integrate 2D and 3D visualisation systems in a seamless manner in order to create "Immersive Analytics Environments" as well as to develop immersive visualisation methods for huge statistical and multidimensional datasets.

5.2 Requirements

This section gives an overview of relevant requirements. These are derived from the CoeGSS workflow (Chapter 2), the description of the pilots (Sections 9.3, 10.3, 11.3) and tool interfaces as well as HPC coupling (Chapter 8).

5.2.1 CoeGSS Portal

In order to specify appropriate requirements and objectives for development, we first have to differentiate three kinds of CoeGSS portal users: The public user, the CoeGSS user and the CoeGSS expert user. The public user will be able to browse publicly available project results. She will be able to browse and investigate data files as read-only snippets without the requirements of data accessing nor processing. The CoeGSS user is meant to login to the portal using a dedicated account, to upload data, prepare data sets, define the workflow services and parameters, and to start and monitor the process. The CoeGSS expert user will have the additional option to start interactive sessions, using HPC resources as well as advanced visualisation tools interactively. While the public user will make use of standard web-based visualisation tools from the cases' domains to browse data and analyse files, task T3.3 will support CoeGSS users and expert users with advanced visualisation tools.

CoeGSS User

As mentioned above, the CoeGSS User is associated using the portal software without the need of using optional processing resources like HPC nor dedicated visualisation servers. The CoeGSS user will be able to use domain specific CoeGSS portal services controlled by user management as a user interface as well as an access point for setup, control and monitoring of the whole process (see Section 2 and Figure 5.1).

The CoeGSS Portal provides tools and services to the CoeGSS user to define and setup scenarios starting from a marketplace (see Section 2). In the context of the CoeGSS project the CoeGSS

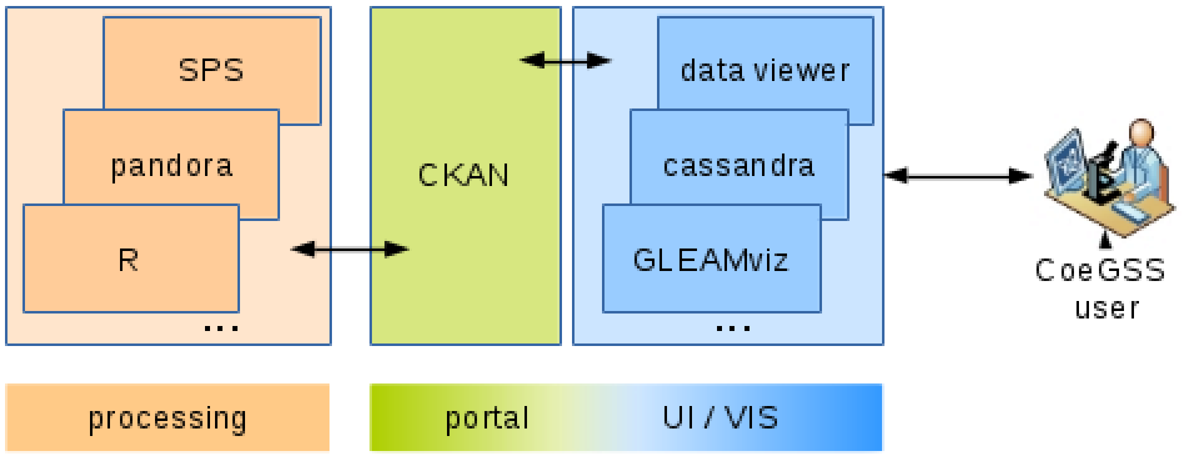

Figure 5.1: System sub-architecture concept for the CoeGSS user

pilots will be the focus in the first portal release. After defining a scenario, the user will be able to initiate batch jobs to process the data. Based on given HPC resource management tools, the jobs will be scheduled and run separately from the portal. Visualisation tools should be capable of providing previews of the setup as well as of input or expected output data as possible without having to start the whole process for the scenario. To enable setup and start of processing jobs, the CoeGSS Portal software is meant to interface various visualisation and pre-processing tools, which have to access the CoeGSS data management system. The first tools focused on, are those used or proposed by the pilots (see Section 5.2.2).

CoeGSS Expert User

The CoeGSS expert user will have the additional option to use interactive services like interactive visualisation and on-demand data processing. This requires availability of corresponding processing and visualisation capabilities, as well as fast access to the data, to respond to user input within appropriate time or real-time if using interactive virtual environments, for instance. Access and resource management is essential, to have the resources on-hand during ongoing work. Furthermore, it is crucial to use HPC resource management tools to integrate the CoeGSS workflow into common HPC centre procedures. Use cases for the CoeGSS expert user can be divided into remote or local usage of visualisation applications. In this context, this refers to where the processing and rendering is actually being done. Remote usage of applications tends to run directly on a HPC or visualisation environment, which handles and processes the relevant GSS data bases and data sets respectively. A user session could make use of a client-server architecture or a desktop sharing system (e.g. VNC, remote graphics) for instance. Various approaches and solutions are known and well established, for example, in engineering[14] or scientific visualisation[15].

In order to be able to investigate and analyse the data output offline or unattached to HPC processing capabilities respectively, the visualisation system should be able to run on local resources (see Figure 5.2). Especially when using virtual reality environments like CAVEs, this is mandatory due to the need of low-latency user interaction and rendering capabilities within multi-screen or high resolution environments. This can only be done with a subset of the GSS data or pre-calculated, pre-processed data prepared for visualisation to provide needed rendering performance at high framerate. Even though running on virtual environments, the

[14] http://www.cospaces.org/
[15] https://www.hlrs.de/solutions-services/service-portfolio/visualization/covise/features/

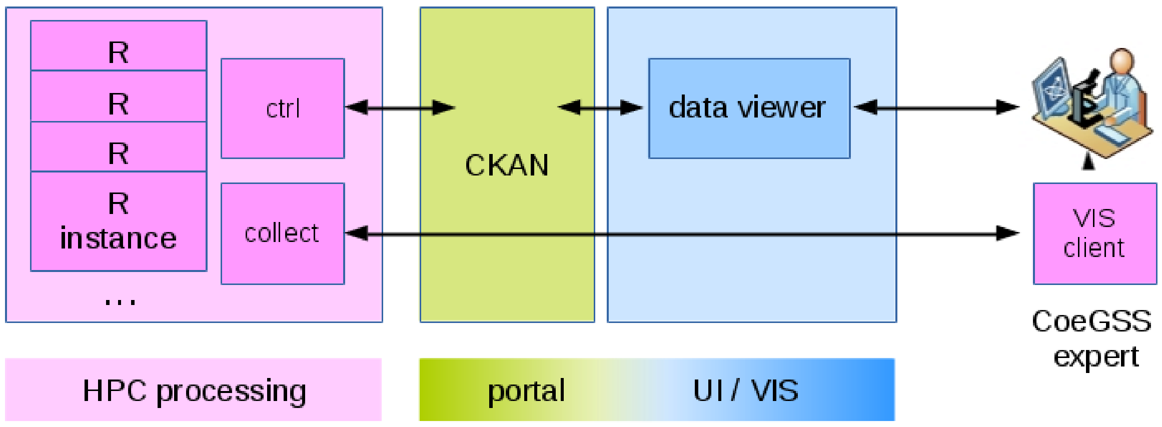

Figure 5.2: System sub-architecture concept for the CoeGSS expert user

visualisation system should provide interfaces to methods or tools for additional data management, and to high performance data analysis tools, which can be started separately. The virtual environment should then integrate a 3D-environment with statistical output of the domain specific GSS tools. This will support the use case "Immersive Analytics Environments" as proposed to be used by experts and analysts to help in the detailed analysis of complex data sets and combining 2D and 3D content in a single environment.

The SIS will be performed on HPC systems coupled with data management systems in order to efficiently provide input data, store processed data and support processing relocation in case of breakdown (see Section 5.2). The visualisation system should not only provide tools for analysing and interpreting the resulting data, but should also assist during design phases and provide visualisation while simulations are running. This will require in-situ visualisation techniques to have access to GSS simulation runs while being processed on HPC resources.

5.2.2 The CoeGSS Pilots

The GSS SIS will be one of the main tools used in CoeGSS, in particular, within all the pilot projects (see Section 5.2), and it is mandatory to provide an interface. However, in the individual pilots, some requirements have to be handled separately.

Health Habits

Within this pilot two visualisation tasks are central. A tool is needed to visualise the results coming from the GSS models as results of spatially explicit stochastic simulations, showing the dynamics of a contagion process over time and space (see Section 9.3). Visualisation tools as the one used by the GLEAMviz simulator have been developed. Specific features may have to be added or the tools have to be integrated into a visualisation framework to fulfil the needs of the pilot. The second task should focus on visualisation of a large parameter phase space coming from processed synthetic population models. This should enable exploration of a large set of parameters through a visual interface.

Green Growth

Tools and applications used in this pilot will mainly make use of storing data in HDF5 file format. Many GIS-Tools (e.g. QGIS) are capable of importing data sets stored in HDF5 file format, but gives only access to a subset of the results as it ignores the agent specific details, for instance (see Section 10.3). For exploring the entire data sets, a visualisation system is needed, which allows the visualisation of GIS data coupled with statistical output as well as investigation of model progress time-series. This should allow the user to compare multiple runs of the model easily.

Global Urbanisation

Using 2D and 3D visualisations might be useful for this pilot to provide a feeling of realism when browsing and investigating the output data (see Section 11.3). However when observing results of city simulations a major stake is also to help reach high level insights from possibly multi-variant, multi-scale, very detailed simulations results.

5.2.3 Tool Interfaces

A general but important requirement for the visualisation system is an open API, which enables users and developers to modify and extend functionality, to read or process data in a specific way, as well as to enable batch processing of huge data sets.

The R Project

R is a language and environment for statistical computing and graphics[16]. It is a GNU project which is similar to the S language and environment developed at Bell Laboratories by John Chambers and colleagues. R provides a wide variety of statistical and graphical techniques, and is highly extensible. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes an effective data handling and storage facility, a suite of operators for calculations on arrays, in particular matrices, a large, coherent, integrated collection of intermediate tools for data analysis, graphical facilities for data analysis and display either on-screen or on hardcopy, and a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities. The visualisation system should interface R libraries, so that the R functionality can be used interactively within the visualisation environment.

Synthetic Information System (SIS)

The visualisation system should be capable of interfacing with the SIS to read and evaluate the data for further analysis of achieved results.

5.3 Solution: The CoeGSS Visualisation Toolbox

There is a tidal wave of big and complex data across many domains. Several research directions are concerned with the development of methods to support the analysis of such data, including machine learning, information visualisation, data analytics and human computer interaction. However, there is still a major gap to fill: how can analysts get immersed in the data to provide more natural ways for data exploration, data analysis and collaboration?

Immersing people in their data does not necessarily involve 3D or stereo display. In this concept (Figure 5.3), analysts work in a purpose-built, collaborative data analytics room. Interaction technologies like pen, touch and gesture control allow them to interact directly with their data and collaborate in a more egalitarian way than keyboards and mice, which tether individual users to a desktop. Immersive Analytics will systematically research how many kinds of emerging interaction technologies can be harnessed to engage and enable people to work together to better understand data.

[16] `https://www.r-project.org/`

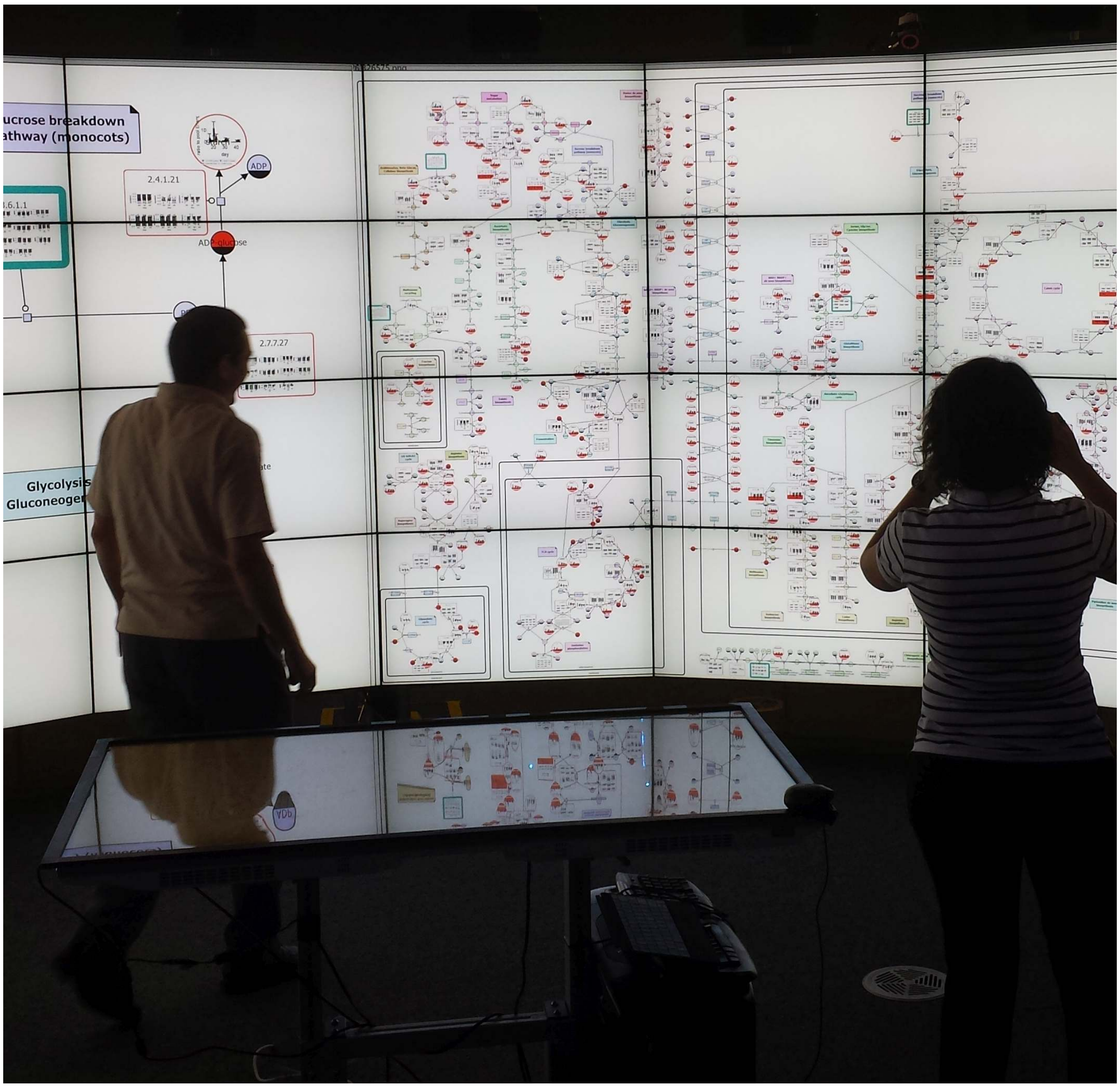

Figure 5.3: ContextuWall at Immersive Analytics, Monash University

The CoeGSS Visualisation Toolbox aims to support partners and users with visualisation solutions capable of reading relevant file formats as well as providing visualisation tools and modules for high-resolution, multi-dimensional, multi-scale data sets while providing interactive access to simulation and processing tools behind the visualisation. At this stage of the project, not all requirements can be formally described in detail and some are still under ongoing discussion in preparation for the Technical Roadmap Meeting in April 2016. Due to that reason the development and deployment of the CoeGSS Visualisation Toolbox will be an agile process focusing on partner needs in the first instance.

5.3.1 Virtual Reality (VR) and R

In recent years there has been significant advances in the development of new technologies for human-computer interfaces. In particular, technologies for natural user interfaces, virtual and augmented reality devices have progressed very quickly to provide very engaging and immersive experiences. The industry, in particular the entertainment industry, starts to adopt, develop, and disseminate basic concepts. As a consequence, tools and methods like voice- and gesture-based control as well as 3D visualisation are becoming more and more part of everyday life, and corresponding devices are becoming available and affordable for small businesses and the general public. Fundamental research is done to further expand the frontier by inventing more sophisticated approaches, and by improving usability and efficiency of available concepts. Immersive environments like a CAVE can make use of ultra-high resolution technology, and combine 2D and 3D visualisations to allow users to immerse themselves into computer generated scenes.

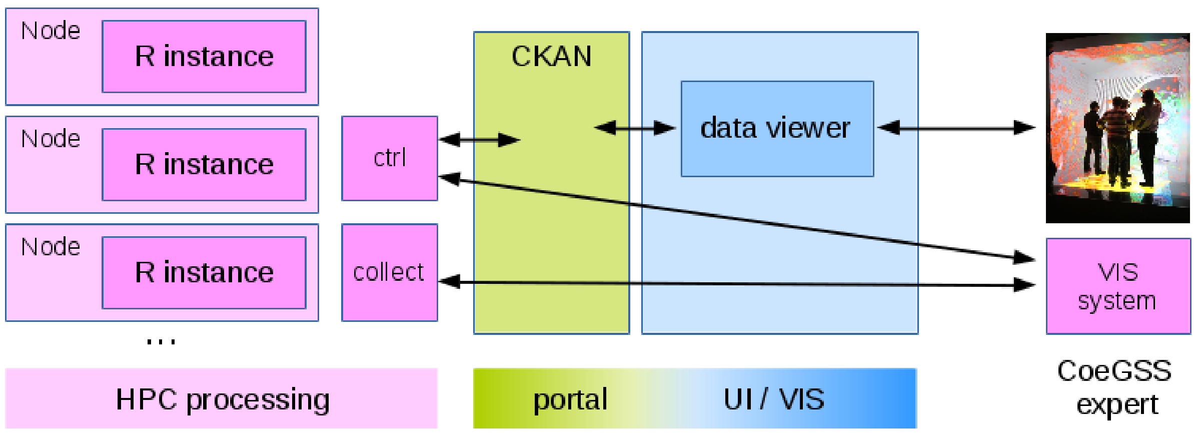

Figure 5.4: HPC-R-VR coupling concept

With coupling the R-Project to the CoeGSS Visualisation Toolbox a new facet of data analytics will be created: emerging natural-user-interface and augmented-reality technologies for real-world analysis of data. With this approach of immersive analytics, partners and users will be capable of getting deep insights from complex, huge data sets by creating more engaging experiences and seamless workflows for data analysis applications.

The ability to move inside the data with new display devices, and to interact with the data representation in various direct and embodied ways, allows the user to create a richer experience that may lead to better understanding and finally deeper insight. For that to happen, the visualisation and interaction concepts need to allow a faithful representation of the data and functionality provided by R's processing algorithms.

Tasks and challenges regarding the coupling of Virtual Reality (VR) to R will be:

- Investigate potentials and challenges of immersive analytics for research and commercial applications for design requirements of the coupling.
- Investigate how existing interaction models and techniques can be adapted to immersive analytic environments, and where completely new approaches are necessary. Formulate guidelines (based on this investigation) for the use of such interaction models and techniques in immersive analytics.
- To investigate the system behaviour of R running in HPC resources while coupled to a virtual reality environment
- To explore the design space for immersive analytics for effective collaborative data analysis in various forms, e.g., distributed or local, synchronous or asynchronous.
- Initial design of open-source tools and plug-ins for supporting immersive analytics using the R project.

5.3.2 CKAN Interface

CKAN CKAN.org is a data management system that makes data accessible by providing tools to streamline publishing, sharing, finding and using data. CKAN is aimed at data publishers and provides a streamlined way to make data discoverable and presentable. Each dataset is given its own page with a collection of metadata, making it a searchable resource. Important features in this context are data storage and visualisation. For example, CKAN provides basic data previewing tools to display data on a graph, choosing the variables on the axes and comparing a number of variables by graphing them together on the same y-axis. It can handle mapped data, image data and table views.

Using CKANs API the portal software can interface[17] the R project. This is meant to be the entry point for further development aiming to implement the conceptual architecture (Figure 5.4).

[17] `https://cran.r-project.org/web/packages/ckanr/`

6 Domain Specific Languages (DSLs)

6.1 Introduction

The overall goal of Task 3.4 on Domain Specific Languages (for the full three year period) are the following according to the DoW:

- Analysis of GSS synthetic populations used in case studies and selection of high-level concepts that make up the DSLs
- Specifications of concepts and their combinations in the form of property-based tests
- DSLs for rapid construction of GSS synthetic populations
- Property-based testing of GSS simulations built using the DSLs

For the Centre of Excellence on Global Systems Science it will be important to set up synthetic information systems (SISs) quickly. The nature of GSS problems [Jaeger et al., 2013] suggests that there will not be a universal SIS. Rather, the situation as reflected by the pilots is that there will be many tools, used for building different kinds of SISs for different purposes. That means that there is a certain tendency towards *fragmentation* and lack of reuse, which must be kept under control. For example, the pilots will generate different synthetic populations using different tools (and using different data, although with a certain amount of overlap). Therefore, we set as a goal being able to reuse the various data sources in different SISs. To achieve this, we need a *specification* of the synthetic populations, to serve as the basis of a domain-specific language (DSL) for translating from one SIS to another. Such a DSL would be a first step towards a front-end for generating GSS synthetic populations in a uniform way, giving a uniform interface to the various tools used in the Centre.

Another important requirement for the SISs used in the Centre is that of “test, verify, and validate the results” [D4.1, p. 9]. Conceiving and implementing tests for complex software systems is hard, and it is desirable to reuse good tests across the Centre’s SISs. Because of the heterogeneity of the SISs, it is unlikely that the *code* for the tests will be very reusable. However, if we focus on *property-based testing* [Claessen and Hughes, 2000], we will be able to reuse the *design* of the tests. Moreover, property-based tests provide a form of specification of the components to be tested, and therefore will increase the reusability of these components, in turn leading to quicker development times for the Centre’s SIS. Property-based tests are also a natural way to connect the contributions of the various tasks, for example, by using components developed by Task 3.2 for validated numerics in order to validate the new software proposed by Task 3.6 to be used in the SISs of the Centre.

6.2 Requirements

The three pilot studies of the Centre have started their activities with a summary of their initial plans and requirements, as outlined in D4.1. Even though the pilot studies have differing requirements with respect to the synthetic populations, we aim at reusing significant parts of raw data and synthetic populations to avoid duplication of effort. For example, both the *Health Habits* the *Global Urbanisation* pilots need human population data, even though their granularity requirements might be different. Similarly, both the *Global Urbanisation* and the *Green Growth* pilots need road network data and car ownership data [D4.1].

At the same time, every pilot has additional synthetic population requirements specific to it. For example, only the *Health Habits* pilot needs the data about people's health. Therefore, we highlight the need for creating a mechanism for reusing data that can be used by all pilots while allowing for custom additions and modifications to it.

It should be emphasised that this arrangement should not *force* the pilots to use any particular set of data, but rather provide them with greater flexibility by allowing them to reuse the data that they find useful. For example, the *Health Habits* pilot starts with the the Socio-Economic Data and Applications Centre (SEDAC) database [D4.1], and is later going to include other sources. What exact data sources are used will depend of their suitability and the quality of data, which is difficult to predict in advance.

In order to provide tools for generating synthetic populations for the pilots we intend to reuse as much as possible from existing packages, such as `simPop` or `SynthPop` [D3.1].

Generating synthetic populations that satisfy differing requirements and are based on diverse sources of data requires having a description of both the constraints that the synthetic population must fulfil and the structure of input and output data.

This can be realised by a domain-specific language for specifying relations between data and structure of synthetic populations, which is an important part of the workflow described in Chapter 2. With such a DSL, we can implement translations between the formats used by the various pilots, enabling the reuse of the static parts of the various SISs, but also laying the foundation for reuse of the dynamical components. Since these components are written in different programming languages, within different environments, it is likely that the easiest way to combine them is via the data they produce and consume. Therefore, a tool that can translate between the Centre's data structures will also lead to a framework of components that can be used for the rapid production of SISs.

A uniform data description is also needed for Task 3.3, visualisation (Chapter 5); in turn, the requirements of visualisation can influence the interfaces of the Centre's SISs.

A common description of the static aspects will also contribute to the task of HPC-GSS system configuration selection (Chapter 8), since the synthetic population that is used is a major factor in the choice of configuration.

Finally, translating from one synthetic population format to another can facilitate the application of data analytics methods briefly presented in Chapter 4, which would otherwise need to be tailored for each application separately.

Apart from reusing synthetic populations, the dynamic aspects of the multi-agent models should also be reused between the pilots. The pilots will implement the simulation using framework for agent-based models, such as Pandora (see Chapter 10). Similarly to synthetic populations, the dynamic aspects of the pilots will share some common features, while also containing significant differences.

We now turn to the other common requirement, that of testing. Testing is a general requirement for all pilots [D4.1, p. 12]:

> [. . .] testing, verification, and validation of results are necessary in any modelling activity. However, given that CoeGSS pilots will use new data sources for large synthetic populations, there is a need to undertake these steps very carefully, and discover requirements (e.g. new methods) while entering this new field.

The testing referred to here is a test of the model and of the implementation of the model. It complements the system-level testing described in Chapter 3 (*Enhanced Reliability and Scalabil-*

ity), where the focus is on the capability of a software system to recover from faulty hardware, irrespective of whether the model implemented by that system is correct or not.

The importance of testing is highlighted by the lessons learnt from the project ILUMASS, a European project sufficiently similar to CoeGSS that ran into unexpected difficulties in delivering its software. The obstacles encountered by that project should be closely examined in order to limit the risk of failure of this project. Among the problems experienced during ILUMASS, inadequate testing is listed prominently [D4.1, p. 14]:

> Reasons for the failure of the project include [. . .] inefficient testing procedures (running long simulations on the whole study area only to recognise a small but critical error at the end) [. . .] This underscores some of the points made above: testing procedures need to be accurately developed [. . .]

The "accurate development" of such procedures is made more difficult by the fact that we will be testing genuinely new methods and models. As D4.1 points out in the context of Task 4.1 (p. 17):

> The definition of such models will require an extensive research effort to identify relevant parameters, calibrate them on real data, and perform numerical simulations.

Finally, the implementation of methods described in Chapter 4 (*Data Analytics*) will also require validation, which can be partially addressed by testing. In addition, testing, verification and validation of numerical methods is a major concern of Task 3.2 (Chapter 7), where the emphasis is on the accuracy of numerical methods.

6.3 Gaps

We identified three primary gaps. The first gap is the lack of reuse of synthetic population data from one pilot to another. While the pilot on *Health Habits* focuses on an existing synthetic population based on data that the team has successfully used in the past, the other pilots are, at least currently, not in a position to reuse significant parts of this data with little effort.

The second gap is connected to implementing the dynamic simulation of different pilots in a way that avoids duplication of effort. Reusing parts of models implemented using agent-based frameworks is difficult as implementations tend to consist of interdependent parts.

The third gap is related to the topic of testing. There is no common testing framework shared by the pilots; in fact, it is unclear whether any of the pilots use a testing framework, or how the testing, identified as such an important requirement of all the pilots, is to be done at all.

In particular, software used for testing in one pilot cannot be reused in other pilots. Thus, the effort to develop effective tests and implement them has to be repeated in each of the pilots.

An interesting gap is identified both in Chapter 10 (*Green Growth*) and in Chapter 11 (*Global Urbanisation*). This is a gap between the uncertainty in models and implementations and the analysis of uncertainty carried out in Task 3.2 (Chapter 7). As we will see in the next section, property-based tests provide one way of closing this gap.

6.4 Solutions

In order to enable the reuse of data for building synthetic populations, we plan to implement software tools that translate between the various data formats. This translation involves a low-level conversion of data representation, such as from comma-separated values to C++ structures, but is driven by higher-level, unifying abstractions. To explain this, consider the following analogy: translating from a quantity expressed on the Fahrenheit scale to the Celsius scale involves a low-level linear transformation; but it is based on the abstract concept of temperature and the sometimes complex ways of measuring it.

In the same way, we need to discover, analyse and classify the abstract concepts that characterise GSS synthetic populations. These will form the core of a DSL for GSS synthetic populations, which can then be used to build GSS synthetic population *generators*.

As the pilot projects come up with specific definitions for their synthetic populations in the coming months, their common data sources will be specified, as well as the required mechanisms for reuse. We expect this process to involve several iterations, in which the pilots adapt their specific requirements in response to the proposed extension mechanisms.

The plan for reusing the dynamic part involves the pilots defining first versions of their agent-based models, and evaluating whether it is possible to reuse any common functionality across different pilots. The results will contribute to the creation of the DSL for the specification of the Centre's agent-based models.

In order to create an efficient testing environment for the Centre's SISs, we plan to develop property-based tests. Currently, most testing in scientific computing is example based. A number of input/output samples are chosen by the programmers. If the implementations produce the desired outputs for the respective inputs, then they are considered correct. There are several problems with this approach:

- another programmer, on seeing the tests, cannot in general tell what property is being tested. In our case, that means that the input/output samples cannot, in general, be used to test a similar implementation, i.e., there is no sharing of tests
- it is often difficult to select the best input/output samples, particularly so in the case of new methods and models. Moreover, if the programmers responsible for the implementation are also those who choose the tests, then errors made in the implementation are likely to be reflected in errors in the choice of the test. A programmer who has neglected an important factor in the implementation is likely to neglect it again in the testing phase.

By contrast, in property-based tests the programmer focuses on expressing the properties that the implementation must satisfy, and the computer will then automatically generate test cases that attempt to "break" the implementation. Since the cases are not chosen by the programmer, they are not subject to the programmer's "blind spots". Most importantly, the tests do specify the properties that are being tested and which characterise the implementation. Thus, they serve as a specification for the implementation; similar implementations will be testable in the same way. Specifications of the components used in the Centre's SISs is an important step towards creating a common language and a scientific platform for rapid modelling with SISs.

Property-based tests provide a way of integrating the contributions of Task 3.2 in the work of the pilots.

To see this, consider the following example, which picks up a theme of Chapter 7. The property that must be satisfied by an optimisation method is that the result is optimal. In an example-based approach, the programmer will choose a number of test functions for which the optimal result is known, and check to see whether the implementation reproduces them. In general, this leads to choosing simple functions as test cases, or, even if more complex examples are available, they are not necessarily representative for the tasks that the optimiser will face in the new GSS applications.

In a property-based testing approach, the optimality property will be specified. Test cases, in this context representing functions, will be automatically generated. The optimality property is then tested by sampling the domain of the functions to be optimised and checking if the results improve the solution given by the implementation. While this is usually a better idea than just testing for some simple functions, we are still faced with a problem of *coverage*. The number of samples required for a confident validation of the implementation may be prohibitive.

The software developed in Task 3.2, being based on validated numerics, offers a standard against which the methods implemented in the pilots can be measured. For an optimisation task, validated numerics produces a set (represented as a union of intervals) in which the solution is guaranteed to lie. Instead of testing the optimality of the implementation by a potentially ineffective sampling process, we can now test it by checking that the solution lies in the set produced by the validated numerical method.

7 Representing uncertainty of computations and modelling

7.1 Introduction

The main goal of task 3.2 is to ensure the validation and correctness of the software components developed for GSS simulations.

To achieve this, we will use cutting-edge research on type theory and functional programming in order to implement software components to deal with the uncertainties that arise from imprecision in computations (floating-point errors) and from uncertainty in data (imprecision in measurements).

One important component in the CoeGSS workflow is the generation of customised synthetic information systems for GSS applications. The generational process involves software components to initialise, adapt and evolve the agents. In task 3.2 we focus on the numerical methods used in the generational process for global optimisation needed to find best-fit parameters for statistical models like Iterative Proportional Fitting (IPF) or simulated annealing.

By implementing techniques of validated numerics that can be used either directly in the generational process to confine the search space or indirectly to create test environments and thus improving the quality of other GSS models or tools (such as visualisation or data analysis) we will ensure validity and correctness for our tools.

7.2 Requirements

As a common requirement for all pilots, D4.1 states that the capabilities of existing frameworks for SIS (Synthetic Information System) should be extended and enhanced by developing new methods for deriving relationships and activity patterns for agents. While most existing synthetic populations are confined to certain countries we aim to develop synthetic populations of fine granularity on a global scale where existing generation methods might become inefficient or not accurate enough.

The authors of D4.1 emphasise the importance of testing and calibration, especially in the context of complex agent based modelling systems (ABMs) such as those required by an SIS. All pilot studies refer to the need for assessing *scenarios* as a way to validate a model against real data.

Based on D4.1 and in the order of the workflow as it is described in Chapter 2 of this document the following list describes in more detail those requirements of WP4 which are related to numerical methods.

1. *pre-processing of data* To feed the models several data sets are needed. However, the quality and availability of data varies. Therefore, pre-processing of data will be required. The numerical methods involved here are spatial and temporal interpolation to interpolate missing information [D4.1, p. 11].

2. *from data to synthetic populations* The standard approach for building synthetic populations consists in "merging aggregate data from a source covering the whole population with disaggregated data from a sample in order to get a disaggregated data set for the population of interest" [Beckmann et al., 1996].

In general there will not be a single source of disaggregated data describing the agents of interest. Instead there will be a number of such sources (seeds) with different sample sizes and granularities. What kind of surveys are used as sources for aggregated information depends on the use case. For instance the health habits pilot will use relevant health statistics.

A SIS requires "creating sets of synthetic populations with realistically statistically distributed characteristics' values", moreover "with realistic distributions not only of single characteristics but taken as a set, together i.e. reflecting correlations" [D4.1, p. 29]. To reproduce the statistically correct representation and to generate a reliable synthetic population various sampling procedures and techniques of combinatorial optimisation like hill climbing, IPF and Monte Carlo methods have to be used.

3. *modelling* One important step in the modelling of dynamic systems is the sensitivity analysis of the models. When collecting data from different sources "The questions of which relationships and activity patterns of the agents are the relevant ones . . . requires the development of new methods " [D4.1, p. 10].

   The aim is to "identify relevant parameters and to adjust them by performing numerical simulations on real data" as it is described in [D4.1, p. 17] for the health pilot. Later on in section 4.3 the use of optimisation methods is requested: "find optimal parameter values with an optimisation algorithm" [D4.1, p. 41].

   The section on the urbanisation task points out that the submodels "can be interdependent, over statistic data or over dynamic coupling".

4. *quality assurance* As mentioned above some steps of the generational process involve the interpolation or estimation of missing information. D4.1 highlights that "it is necessary to track the quality of these estimates throughout all pre-processing steps for quality assurance" [p. 11].

5. *analysing outputs* The simulations can be seen as probabilistic processes, even with the same parameters they yield different outputs each time they are run. These different outcomes have to be analysed together. This will be done by using R and other statistical tools to extract results from sets of simulations.

7.3 Gaps

7.3.1 Validated numerics

Every numerical calculation on computers is not done with real numbers but with a finite stock of machine representations.

As a consequence numerical computations are almost never carried out in a mathematically precise manner. The results produced are not exact but rather approximations that are usually but not always near the true ones. Although IEEE standards ensure the precision of single operations it is hard to track the accuracy of an algorithm over a whole series of computations. To analyse an algorithm with regard to its accuracy would involve to observe the deviations from every single floating point operation throughout the whole computation! With a complex system involving very long computer runs, this is out of reach.

If new software has to be developed for creating synthetic populations the whole cycle of software creation is necessary, including specification and testing. This includes several numerical procedures.

The gap here is the unknown accuracy of the software components in use. To know the precision of the implemented algorithms is in particular important for the creation of test environments.

7.3.2 Non-linear optimisation

For the creation and adaption of SIS it is necessary to employ several optimisation techniques. The optimisation problems that have to be solved e.g. for parameter identification, or creating realistic distributions for agents, are non-linear. One consequence of this is that, in contrast to linear problems there may exist several (local) optima. Although a number of methods for the solution of constrained nonlinear problems are available, there is no general method known to determine the global optimum of an arbitrary nonlinear problem with certainty. With classical local optimsation methods one finds an optimum but it depends on the starting point whether it is a global one. Although they may converge, local methods tend to become stuck at local optima and an extremum elsewhere in the parameter space may be neglected. This might be sufficient in systems with few parameters and agents, but it can be very damaging in an SIS with hundreds of millions of agents.

Global optimisation methods like “simulated annealing” might overcome this difficulty but they rely on the inclusion of random elements. One drawback here is that, even if we are able to prove that the algorithm in eventually will converge to a global optimal solution with probability 1, we usually don’t have convergence results that specify a time limit within which the algorithm is guaranteed to converge (with some high probability, say). Unfortunately testing of optimisation software to find out how precise a computed solution might be is a notoriously difficult problem (see, for example, [Murray-Smith, 2015] and [Dolan and More, 2002]).

We want the results of optimisations to be provably accurate to a certain degree.

The gap here is that neither local nor global optimisation techniques in existence are validated and reliable in that sense.

7.3.3 Efficiency of combined systems

Synthetic information systems as currently conceived are stochastic systems. They are supposed to run under different scenarios, representing, e.g., different policies. This is underlined many times in D4.1 as a source of computational complexity, especially if the scenarios are not *static*, but, as is more realistic, can depend on the evolution of the SIS. The situation is that of a combination between a stochastic system (the SIS) and a non-deterministic one (the scenarios). Keeping the two systems separate means taking a worst-case approach to the computation: the stochastic system is run once for each scenario, ignoring any connections, convergence, or overlap between the scenarios. The gap here is the absence of infrastructure for combining the systems.

7.4 Solutions

To keep track of the accuracy of used numerical methods we will implement a validated numerics library in Idris, a functional programming language [Brady, 2013]. The approach we use here is interval arithmetics as described in [Tucker, 2011]. In interval arithmetics all quantities are treated as intervals, which are propagated throughout a calculation. The final result is an interval that is guaranteed to contain the correct result, starting from the given initial data. Thus we obtain a measure of the precision of our result.

Since Idris is a dependently typed language it allows us to derive the proof of correctness of an algorithm in parallel with its implementation. The implemented numerical algorithms will be used for interpolation and to create test environments for software components of the HPC framework for synthetic populations under development. (Some of the trade-offs between testing and proving are explained by Ionescu and Jansson [2013].) Most importantly, the correctness proofs of our algorithms ensure the correctness of our test data.

A similar approach will be used to address the gap in subsection 7.3.2. By re-implementing optimisation methods like iterative proportional fitting or gradient descent using interval arithmatic we will obtain test environments for standard implementations of optimisation procedures. The further development of the pilots will determine which of the optimisation methods to focus on here. These implementations will produce intervals provably containing a global optimum. If in an acceptable amount of time it is not possible to calculate an interval that is small enough for our requirements we can still use the calculated interval to choose promising starting values for standard implementations.

8 Hardware and software co-design

8.1 Introduction

In continuation of D3.1 where the four essential requirements of the co-design approach have been stated according to Sanders and Stappers [2008],

- Centred around user's needs
- Focused on designing for a purpose
- Longer views and addressing larger scopes of inquiry
- Change of roles in the design process

at this place methods will be specified to close the gaps between the requirements given by the GSS community, represented in CoeGSS by the three use-case providers, and the HPC community. The methods defined in the following sections will be represented as modules in the CoeGSS co-design toolbox. The idea for these modules as well as for the complete co-design toolbox is the one of a growing structure, that evolves along with the GSS co-design process.

To present the initial module layout along with an initial set of solutions to close the most urgent gaps identified from D4.1 the sections "Requirements", "Gaps" and "Solutions" separately presented in the previous chapters are fused together in this one to give an idea about the actual implementation of the CoeGSS co-design toolbox and its initial module layout together with the methods and solutions that are collected in them.

A clear and urgent gap to close is the unawareness of the GSS community about the technical capabilities and limits of present day High Performance Computing (HPC).

Successful co-design requires mutual awareness of the parties participating in the design process. In our case the GSS community has problems communicating with the HPC community and its stakeholders. Therefore, a first step towards the implementation of the CoeGSS co-design toolbox is the specification and implementation of the "User-Awareness Creation Module" (U-ACM). This Module will provide and collect general information related to HPC-hardware and infrastructure as well as information, methods and techniques that enable the high-performance, efficient and also energy aware programming and software development for current and upcoming HPC-hardware and infrastructure.

The second step towards the implementation of the CoeGSS co-design toolbox is the specification and implementation of the three "Knowledge Base Modules" (KBMs),

- Hardware Knowledge Base Module (H-KBM)
- Software Knowledge Base Module (S-KBM)
- Analysis Knowledge Base Module (A-KBM)

which will collect specified knowledge about hardware, GSS software has been executed on, GSS software that has been executed on HPC Hardware along with use case specifications, as well as evaluation results and evaluation techniques that have been applied to both Hardware and Software facilitated in GSS applications.

The KBMs will also serve as a starting point for the two other CoeGSS co-design toolbox modules that have to be developed and that will be specified in the second release of this document:

the “HPC-Community-Awareness Creation Module” (HPC-ACM) and the “Vendor-Awareness Creation Module” (V-ACM). The specification of these modules is shifted compared to the UACM and KBMs, since more clear ideas about the algorithms and software packages used have to be established before a strategy for the integration of Vendors of HPC systems and infrastructure and also more stakeholders from the field of HPC into the co-design process can be developed.

8.2 User-Awareness Creation Module

The U-ACM, as its name suggests, is intended to create the awareness of the GSS user community for HPC. This will be achieved by two main features. First of all the module will host information directly and by reference about current and upcoming HPC systems and their capabilities along with information about the programming and usage of these systems in efficient ways with high-performance. The second feature of the module is intended to transform over time from a collection of educational possibilities in the filed of HPC, like the ones listed in section 6.2.2 of D3.1, towards an education program that integrates existing off-line and on-line courses with courses created from the information provided by the module itself.

8.2.1 System types and their capabilities

As a starting point this module will hold information about typical HPC-System types as they can be found in the HPC-Centres across Europe today. This will include Systems for

- Capability Computing
- Capacity Computing
- SMP systems
- Fast scratch space systems
- Vector Systems
- Accelerator Systems

The information that is provided by this module will be of general kind and should to the largest possible extent be made publicly available and be related to the special requirements of the software used in GSS.

As an initial set of capabilities that will be described in the U-ACM for all system types held by the module, the following topics are targeted:

- Performance per node
- Memory Bandwidth per node and core
- I/O Bandwidth per node and core
- File access rates pre node
- Bandwidth of the internal network interconnect
- Bandwidth of the external network connections
- I/O connections between centre systems
- Visualisation capabilities

8.2.2 Programming techniques

According to D4.1, all three use case providers plan to use or implement codes written in C++. Since, from the HPC point of view, some language features and implementation techniques of object oriented programming languages are rather sub-optimal when code performance is considered, this section of the U-ACM will contain detailed discussions of program examples, compute kernels and algorithmic implementations from which performance penalties are likely to arise.

As a proof of concept for the statements made above the example of a C++ std::vector used in a sub-optimal way, which means in our case not high-performance way, is shown below.

```
// Create vector -------------------------------------
std::vector<double> x, y, r;

// Init vector ---------------------------------------
for (int ii = 0; ii < nn; ++ii) {
  x.push_back(ii); y.push_back(ii); }

// daxpy ---------------------------------------------
std::vector<double>::iterator ity = y.begin();
for (std::vector<double>::iterator itx = x.begin();
                                   itx < x.end()  ;++itx){
  r.push_back(a * *itx + *ity); ++ity; }
```

If the performance of the version shown above is compared to the version shown below, where the available information about the requested number of elements is used to directly construct vectors with the needed size, the performance penalty that is introduced by the usage of the `push_back` method along with iterators becomes obvious.

```
// Create vector -------------------------------------
std::vector<double> x(nn) y(nn) r(nn);

// Init vector ---------------------------------------
for (int ii = 0; ii < nn; ++ii) {
  x[ii] = ii; y[ii] = ii; }

// daxpy ---------------------------------------------
for (int ii = 0; ii < nn; ++ii) {
  r[ii] =  a * x[ii] + y[ii]; }
```

The examples show two implementations of the daxpy routine from the BLAS library[18]. As command line arguments the number of vector elements and the number of repetitions to get a reasonable time span to measure have to be given. If both examples are compiled with g++ 5.3.1 with the -O3 option on an Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz[19] the speed-up of the second version compared to the first one for $1 \cdot 10^6$ vector elements and 10 repetitions is 2.37. For 1000 vector elements and 10000 repetitions the speed-up increased to 6.13 when using the second version.

One can easily see, that such performance penalties are not acceptable once a code is executed, that runs for hours on thousands of cores. On the other hand its is also obvious that such

[18] http://www.netlib.org/lapack/explore-html/d9/dcd/daxpy_8f.html
[19] http://ark.intel.com/products/75460/Intel-Core-i7-4500U-Processor-4M-Cache-up-to-3_00-GHz

penalties can not always be avoided or reduced by simple programming techniques but the example shows what kind of pitfalls can arise from the extensive usage of flexible and “stylish” programming constructs.

To create awareness for these pitfalls on different systems with different programming languages and techniques among the GSS-HPC software developers and programmers is the aim of the programming techniques section of the U-ACM

8.3 Knowledge Base Modules

For the hardware and software KBMs a database configuration is targeted that delivers measures for full GSS simulation tool-chains as described in Chapter 2. This means the KBMs will start of with the evaluations of the Pandora library currently used by GCF for the Green Growth pilot and the GLEAM simulator, which is in its current configuration used by ISI for the simulation of pandemics around the globe. The initial evaluations will be carried out on the systems hosted at PSNC and HLRS.

For the A-KBM a database structure is not targeted as the primary configuration of the module should deliver descriptions in the form of technical reports, tutorials and HowTos about how the measures presented by the H-KBM and S-KBM where derived.

8.3.1 Hardware Knowledge Base Module

The H-KBM Module is intended to be initialised by kernel measurements, delivering details about the features of the HLRS Cray XC40 System Hazelhen used by the Pandora and GLEAM Simulator codes. As far as it can be determined by now from D4.1 the two most reasonable features to start with will be:

1. The capability of the systems to treat large numbers of small files as they are arising from the application of massive parallel serial program workflows like the ones planned by ISI.
2. The performance of the parallel application of the HDF-5 library as it is used for the I/O of the Pandora library.

8.3.2 Software Knowledge Base Module

The S-KBM will be started with the performance numbers obtained by the application of the “Cray Performance Analysis Tools - CrayPat”[20] to the “RandomWalkers example[21]” delivered along with the Pandora library.

8.3.3 Analysis Knowledge Base Module

The A-KBM Module will start with the description of the application of CrayPat to the “RandomWalkers example” delivered along with the Pandora library in form of a HowTo as it is currently done in the CoeGSS-Wiki[22].

8.4 CoeGSS co-design toolbox

In Figure 8.1 the more detailed layout with the newly developed specifications of the above mentioned modules of the CoeGSS co-design toolbox is presented.

[20] http://docs.cray.com/books/S-2376-610/S-2376-610.pdf

[21] https://github.com/xrubio/pandora/tree/master/examples/randomWalkers

[22] http://wiki.coegss.eu/doku.php

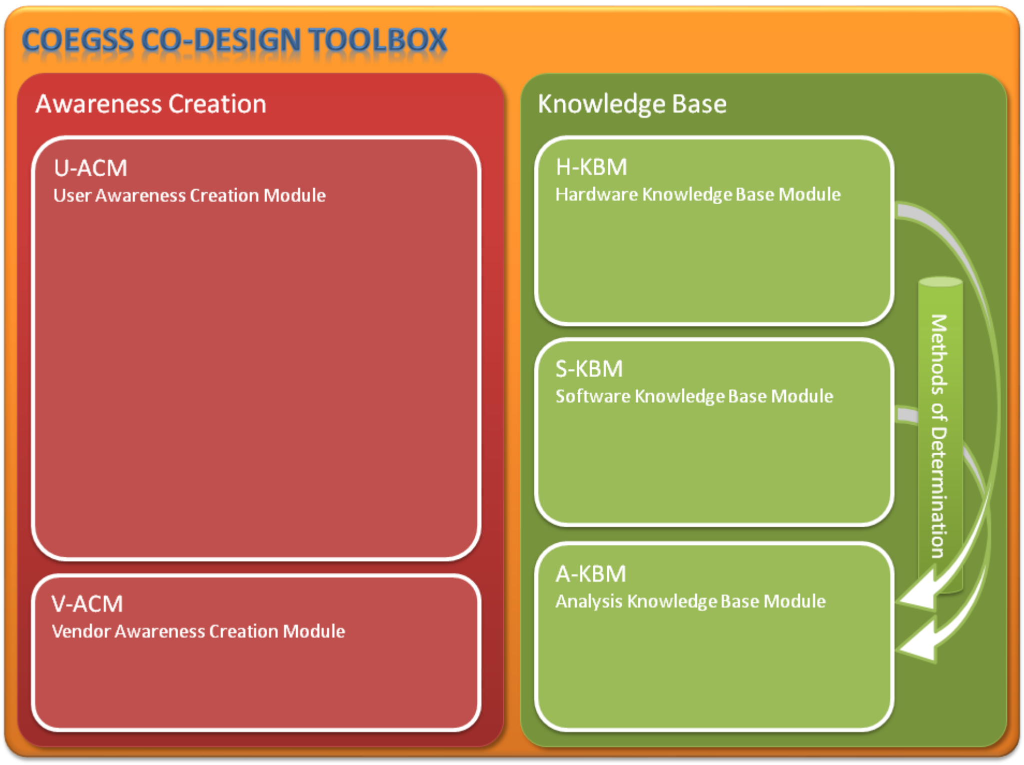
COEGSS CO-DESIGN TOOLBOX
Awareness Creation
U-ACM
User Awareness Creation Module
V-ACM
Vendor Awareness Creation Module
Knowledge Base
H-KBM
Hardware Knowledge Base Module
S-KBM
Software Knowledge Base Module
A-KBM
Analysis Knowledge Base Module
Methods of Determination

Figure 8.1: CoeGSS co-design toolbox

9 Health Habits

9.1 Introduction

Many health conditions are caused by risk behaviours, such as problem drinking, substance use, smoking, overeating, or unprotected sexual intercourse. Monitoring, forecasting and controlling the spread of such behaviours in the general population represents a challenge for policy makers. The main goal of this pilot is to create a Synthetic Information system, integrating large and heterogeneous data sources, that will allow to describe and model the spread of relevant health habits at the population level in Europe.

This chapter is a short reflection on what the methods, tools and mechanisms from WP3 could contribute to the implementation of the Health Habits pilot.

9.2 Data management and fault tolerance

As already identified with partners at HLRS, one key issue to be addressed when deploying agent based models on HPC is the efficiency of the I/O stream.

In terms of input, the synthetic populations of T4.1 are expected to be based on some parsimonious datasets. Demographic information, statistics at population level are needed at this stage, which do not configure as Big Data. No real time data stream will be included in the initial stage of pilot. On the other hand, the size of the output produced by numerical simulations of these models can be significant. Numerical simulations are usually stochastic, which means that many realisations are simulated at once (in the order of thousands). The output is then fragmented in thousands of small files, corresponding to the single realisation and aggregated for each geographic or demographic unit of the model. The output fragmentation can be an obstacle for an efficient simulation process.

Such issues could be addressed within the WP3 by developing/testing some specific HPDA solution. We will experiment with using the Hadoop system in combination with HDF5 containers.

9.3 Visualisation

We foresee two domains where visualisation will be an important task. The first one is the obvious visualisation of model's output. We will need a tool to visualise the results of spatially explicit stochastic simulations, showing the dynamics of a contagion process over time and space. We have developed visualisation tools as the one used by the GLEAMviz simulator, however, specific features may be added to fulfil the needs of the pilot.

The second domain is the visualisation of large parameters phase space. A typical situation is that the synthetic population model will be run by exploring a large set of parameters. Usually, this exploration is not done through a visual interface but only numerically. Developing a visualisation tool for this task could be an important asset to improve the usability of the models.

9.4 Uncertainty

This is a key issue to be addressed when dealing with stochastic simulations and agent based models.

Again, the main source of uncertainty will be the output of the simulations. Usually, results are displayed and visualised as averages and confidence intervals over thousands of models realisations. To compute median and confidence intervals, it requires that all realisations results are stored and analysed at the end of the simulating process.

A DSL could help in this task by specifying what aggregation of the results is wanted and by compiling this to code which only retains the part of the data needed for this computation.

9.5 Hardware and software co-design

We plan to design and code the synthetic information systems without using any specific existing software or developing framework. Some input dataset will be imported from the GLEAMviz simulator, algorithms will be implemented in C/C++ or similar language.

It will be important to make sure from the very beginning that the code is suitable for an efficient deployment on the HLRS. We have therefore started by porting the libraries needed for GLEAMviz.

10 Green Growth

10.1 Introduction

As a first step, the green growth pilot has produced a very much simplified model of the evolution of the global car population, in order to be able to start some tests in the HPC universe. The "synthetic population" that the model is initialised with, for now consists of a number of "brown" and "green" cars for each cell on a gridded world map, obtained by integrating information from various datasets, such as population, streets, cars per 1000 inhabitants per country. In the dynamic model, the evolution of the total number of cars per cell is provided from data. The number of green cars is then determined via a "contagion model": it depends on the number of green cars already present in the neighbourhood of the cell, and on further information such as GDP. This simple model prototype is implemented with the help of the Pandora framework for HPC agent-based modelling. It has been compiled and run at both supercomputing centres in CoeGSS.

10.2 Data management and fault tolerance

10.2.1 Efficient data movement

The green growth pilot uses Pandora, a "a framework designed to create, execute and analyse agent-based models in high-performance computing environments"[23], as modelling platform.

Pandora stores the results of a simulation in different files, using the HDF5 data model. One of those files contains locally aggregated results like e.g. the number of cars in a single cell, whereby a cell is representing a concrete area of the world. Additionally one file is created for every spawned process. Those files contain the values of the attributes of the agents that are assigned to that process. This has the consequence that for a single agent the written attributes values are distributed over several files in the case that the agent is moved to a different process during a simulation. The exact specification of the file formats can be found in the Pandora github repository[24]. The Pandora framework comes with an Analysis module that allows the user to calculate basic statistics generated from the data stored during a simulation. However, in the current implementation all the data must be read into the memory first, and the calculation is done by single-threaded functions. For the agent files a MapReduce solution would be a much better approach. Apache Spark could be useful for this. A possible way to combine HDF5 and Spark is sketched in `https://hdfgroup.org/wp/2015/03/from-hdf5-datasets-to-apache-spark-rdds`.

10.2.2 HPC fault tolerance

Pandora does not support any fault tolerance technique. A checkpointing mechanism could be very useful, as it also allows to reuse simulation calculations e.g. when policies are introduced at an intermediate step in a simulation or the model has a warm-up period. In this case the trajectories of different simulations have a common part at the beginning, so that it is not necessary to calculate this part again for every simulation.

[23] `http://xrubio.github.io/pandora/`

[24] `https://github.com/xrubio/pandora/blob/master/docs/documentation/file_format.tex`

10.3 Visualisation

As already mentioned, the simulation output is stored in HDF5 files. Many GIS-tools like e.g. QGIS can import matrices stored in HDF5 as raster, but this only gives access to a subset of the results as those raster files do not contain the agent specific details. Additionally the GIS-tools ignoring the dynamic character of the rasters, the different time-steps of the same rasters are imported as many uncorrelated static rasters. So for the exploration of full-blown simulation runs, a tool that has the following interactive features is needed:

- Show the dynamic rasters of the Pandora result files as animations, with the possibility to zoom into regions.
- Show an attribute value for a subset of the agents on those animated dynamic rasters (all agents have an x/y position).
- Show time-series of statistical figures based on the raster and agent datasets (e.g. the average income of the agents by country).
- Show raster animations of statistical figures based on the raster and agent datasets (e.g. the average of the agents' income per raster point).
- Read multiple runs of the simulation and allow to compare them easily.

10.4 Domain Specific Languages for SISs

The properties of the synthetic population needed in the model are not determined yet. Currently the idea goes into the direction that the household will make a decision about whether and what kind of car to buy based on the internal attributes (e.g. income, distance to the working place, cultural background . . .), the environment (e.g. infrastructure around the home) and policy measures (e.g. driving regulations in cities).

10.5 Uncertainty

At the current state of the Green Growth pilot, the main source of uncertainty are the data sources. The quality of the data sources fluctuates heavily for the different countries, e.g. for some countries data about the car fleet is totally unavailable.

We plan to do sensitivity, uncertainty, and robustness analysis to explore the reliability of our model. Therefore it is necessary to create experiments that use multiple runs of the model to explore the parameter space. Pandora supports an m^k factorial design, but there exist better designs that are adjusted to the different types of analysis, e.g. the elementary effects method for performing a global sensitivity analysis.

A tool like SimEnv, a "Multi-Run Simulation Environment for Quality Assurance and Scenario Analyses of Models"[25] would be therefore a great help.

[25] `https://www.pik-potsdam.de/research/transdisciplinary-concepts-and-methods/tools/simenv`

11 Global Urbanisation

11.1 Introduction

Pilot 4.3 is intended to be implemented with the CoSMo modelling and simulation platform which automatically generates a C++ based simulation software based on high level specification. This might represent an interesting use case of integrating pre-existing simulation software in the GSS and HPC / HPDA offer. (To protect CoSMo's intellectual property rights, CoeGSS-internal access to the source code is for now covered by non disclosure agreements.)

CoSMo projects are implemented in C++, so even if CoSMo allows for plugins in other languages (particularly Python and Java), C++ friendly options are preferred by CoSMo.

11.2 Data management and fault tolerance

Pilot 4.3 will hold structured data (please refer to D4.1 for details), but network data facilities will be welcome particularly for transport, travel and social networks.

11.2.1 Data management / data analytics

One of the foreseen purposes of pilot 4.3 is to study city models at different granularities (following a problem linked to complex systems), so facilities allowing to aggregate / disaggregate data would be precious. This difference in granularity might be

- spatial: real-estate pricing granularity or precision of travel description,
- temporal: differentiate transport conditions for week days from week-ends, or months following vacation / economic activity or
- conceptual: differentiate populations socio-economically (e.g. profession indifferent, or by sector, or by branch... or class of age vs precise age, ...).

Furthermore, analytics allowing to analyse simulation results and pilot model exploration would be precious (particularly if allowing a link to R for instance). Such analytics might allow to

- calculate basic statistics such as average, standard deviation, ...
- find finer insights on high level indicators (for example, high level correlations between two indicators or applying principal component analysis on a larger set of them)
- or even applying optimisation algorithms to find optimal values of some parameters

11.3 Visualisation

2D and 3D visualisations would be interesting to provide a feeling of realism when simulating the city. However when observing results of city simulations a major stake is also to help reach high level insights from possibly very detailed result simulations and linked to (post-processing) analytics.

Therefore interactive visualisation (in terms of what is observed) allowing the customer to navigate in a profuse set of simulation results possibly at different scales, might be interesting (cf requirement of the role played by visualisation also in the design phase).

In all cases a version runnable on a PC or laptop and not requiring HPC resources would be welcome.

11.4 Domain Specific Languages for SISs

Data for the city pilot will require disaggregating too coarse data.

This might be necessary particularly to map precise locations, or to figure out correlations and relationships between the different static and dynamic (activity patterns) characteristics of individuals, which might further vary over time. Two possible pathways are either to keep data (unrealistically) coarse (e.g. all the persons in the same district holding similar characteristics regarding some point), or else to simulate a random distribution around assumed average and standard deviation values. The first option might miss some realism by eluding effects linked to heterogeneity. The second option raises questions such as ensuring that every random draw respects the aggregate values (average, standard deviation, . . .) precisely enough (i.e. with a sufficiently high number of agents). Furthermore if simulating heterogeneity spatially, it should reflect "realistic" patterns (typically levels of income mostly follow real estate pricing with a certain spatial progressively / continuity and not purely random spatial distribution).

11.5 Uncertainty

Even though this is a very interesting scientific subject, the city pilot has no specific needs or preferences yet in this area.

11.6 Hardware and software co-design

As concerns parallelization, pilot 4.3 will initially seek parallelization of sets of simulations rather than of a single one. Indeed CoSMo's models do not currently allow for parallelism within one simulation. This limitation is an interesting challenge for CoeGSS and in the traditional design perspective the answer would probably be a "closed door" to the HPC centres. But with the system co-design perspective our HPC experts will work together with the GSS experts from the pilot to find ways around the problems. The goal is to find a way to develop a highly scalable version while at the same time protecting the intellectual property rights of CoSMo.

The acquired expertise in this area will broaden the scope of possible use cases for the HPC offer (and the broader the offer the wider the market): customers coming with proprietary, not directly parallell, codes and requesting HPC resources to thoroughly explore its behaviour, while interested in advanced HPDA and GSS expertise to help study and analyse the dynamics of their simulated system.

Indeed Global System Science is not only linked quantitatively to a great number of agents but also qualitatively to the complexity of heterogeneous systems with non linear evolution processes.

The scientific idea of the global urbanization pilot is to provide arguments defending complex modelling and linked HPC needs. Therefore we firstly hope to show, hopefully based on data sets, how a city model with different levels of detail succeeds in capturing (or not) the complex evolution of the reality. Secondly we foresee to calculate high level indicators such as resilience. These two objectives will require many simulations to explore the dynamics of these models to further understand their evolution and compare them (attractors, sensitivity analysis to key parameters, . . .).

12 Summary and next steps

This deliverable (D3.2) is the first release (in project month 6) of a living document of work package 3 (WP3). The next step is a technical roadmap meeting in April (month 7) to plan our developments and synchronise our work for the next few months. We aim to keep updating the living document and produce an internal snapshot in month 12 in connection with the M12 general assembly and in month 18 in connection with deliverable D3.5 “Documentation and software on new methods, tools and mechanisms for Release 2 of the Portal”. The second public release in month 21 will be D3.3 and the third and final release in month 31 will be D3.4.

In CoeGSS, WP3 supports the pilots (and, later on, external partners) with research and development of methods, tools and mechanisms for high performance simulations of global systems. We have identified a common “CoeGSS workflow” which for a particular use case (like our pilot studies) entails the following steps:

- collect, clean and store input data,
- use DSLs to specify a suitable synthetic information system (SIS),
- implement (create) the SIS,
- run the simulation,
- analyse the resulting data,
- visualise the results.

Based on the needs of the pilots on Health Habits, Green Growth, and Global Urbanisation we have identified requirements; gaps as compared to the state-of-the-art; and proposed solutions for each of our tasks: Enhanced Reliability & Scalability, Data Management / Data Analytics, Visualisation, Domain Specific Languages, Types for uncertainty, and Co-design.

The theme for the next few months is scalability and optimisation: we will push the existing software to its limits to identify the bottlenecks where improvements are needed. At the same time we will carefully validate and benchmark our models — there is no need to arrive at the *wrong* answer fast.

References

R. J. Beckmann, K. Baggerly, and M. D. McKay. Creating synthetic baseline populations. *Transportation Research Part A::Policy and Practice*, 30(6):415–29, 1996.

E. Brady. Idris, a general-purpose dependently typed programming language: Design and implementation. *Journal of Functional Programming*, 23:552–593, 2013. ISSN 1469-7653. doi: 10.1017/S095679681300018X. URL `http://journals.cambridge.org/article_S095679681300018X`.

K. Claessen and J. Hughes. QuickCheck: A lightweight tool for random testing of Haskell programs. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*, ICFP '00, pages 268–279. ACM, 2000.

E. D. Dolan and J. J. More. Benchmarking optimization software with performance profiles. *Mathematical Programming*, A(91):201–213, 2002.

N. L. Hjort et al. *Bayesian nonparametrics*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 2010. ISBN 978-0-521-51346-3. URL `http://opac.inria.fr/record=b1130219`.

C. Ionescu and P. Jansson. Testing versus proving in climate impact research. In *Proceedings of the 18th Workshop Types for Proofs and Programs (TYPES'11)*, volume 19, pages 41–54, 2013.

C. Jaeger, P. Jansson, S. van der Leeuw, M. Resch, and J. D. Tabara. GSS: Towards a research program for Global Systems Science. `http://blog.global-systems-science.eu/?p=1512`, 2013. ISBN 978.3.94.1663-12-1. Conference Version, prepared for the Second Open Global Systems Science Conference June 10-12, 2013, Brussels.

M. Lawenda, E. Richter, W. Schotte, C. Ionescu, R. Schneider, and D. Dubhashi. D3.1 – available methods, tools and mechanisms. Technical report, PSNC, 2016. URL `http://coegss.eu/deliverables/CoeGSS_D3.1.pdf`.

D. Murray-Smith. *Testing and Validation of Computer Simulation Models: Principles, Methods and Applications*. Simulation Foundations, Methods and Applications. Springer International Publishing, 2015. ISBN 9783319150994.

CKAN.org. *CKAN — The open source data portal software*. Open Knowledge Foundation. URL `http://ckan.org/`. API documentation at `http://docs.ckan.org/en/latest/api/`.

E. B.-N. Sanders and P. J. Stappers. Co-creation and the new landscapes of design. *CoDesign*, 4 (1):5–18, 2008.

W. Tucker. *Validated Numerics — A Short Introduction to Rigorous Computations*. Simulation Foundations, Methods and Applications. Princeton University Press, 2011. ISBN 978-0-691-14781-9.

S. Wolf, D. Paolotti, T. Michele, M. Edwards, S. Fürst, A. Geiges, A. Ireland, F. Schütze, and G. Steudle. D4.1 – first report on pilot requirements. Technical report, CoeGSS, 2015. URL `http://coegss.eu/deliverables/CoeGSS_D4_1.pdf`.