CoeGSS

Centre of excellence

D3.1 - AVAILABLE METHODS, TOOLS AND MECHANISMS

Grant Agreement	676547
Project Acronym	CoeGSS
Project Title	Centre of Excellence for Global Systems Science
Topic	EINFRA-5-2015
Project website	http://www.coegss-project.eu
Start Date of project	October 1, 2015
Duration	36 months
Due date	31 January 2016
Dissemination level	Public
Nature	Report
Version	1.0
Work Package	WP3
Leading Partner	PSNC (Marcin Lawenda)
Authors	Eva Richter, Wolfgang Schotte, Cezar Ionescu, Ralf Schneider, Devdatt Dubhasi, Marcin Lawenda
Internal Reviewers	E. Richter , B. Koller, J. Nieto
Keywords	Methods and tools for GSS, DSL,
Total number of pages:	82

Version History

	Name	Partner	Date
From	**Marcin Lawenda**	**PSNC**	**20.10.2015**
First Version	**Eva Richter, Wolfgang Schotte, Cezar Ionescu, Ralf Schneider, Marcin Lawenda**	**UP, USTUTT-HLRS, CHALMERS, PSNC**	**25.11.2015**
Second Version	**Eva Richter, Wolfgang Schotte, Cezar Ionescu, Ralf Schneider, Devdatt Dubhasi, Marcin Lawenda**	**UP, USTUTT-HLRS, CHALMERS, PSNC**	**7.01.2016**
Reviewed by	**B. Koller, J. Nieto and E. Richter as consolidator**	**UP, USTUTT-HLRS, ATOS**	**25.01.2016**
Approved by	**ECM Board**	**UP**	**31.01.2016**

Abstract

This deliverable provides an evaluation of available methods, tools and mechanisms which can be integrated off-the-shelf within the Centre of Excellence for Global Systems Science. GSS simulations are processes which consist of many stages like data acquiring and storing, data interpretation and analysis, computation and visualisation. In each stage a whole set of techniques and tools is needed to provide automated and reliable processing. We start with the analysis of efficient data management software and solutions used for fault tolerance. The next chapter introduces the reader into the world of remote and immersive visualisation systems potentially used within the project to explore, investigate and analyse GSS related data. Then, we present a brief evaluation of available software for creating and running GSS synthetic populations, and for designing and implementing domain-specific languages (DSLs). In the following chapter we present type theoretical concepts that could be used to represent different kinds of uncertainty including a short overview of existing implementations. Subsequently a short introduction about the co-design approach in contrast to the classical design approach is given.

Table of Contents

1. Introduction

The definition of Global Systems Science (GSS) is related to systems struggling with the complex problems of global reach. The most popular examples of GSS deal with energy, water and food supply systems, global financial and city systems and globally spreading diseases. In the CoeGSS project we are mostly focused on three domains: **Health Habits** (the global diffusion of health-relevant habits such as smoking, overeating or physical exercise), **Green Growth** (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 **Global Urbanisation** (systemic impact of infrastructure decisions on key urban performance indicators such as congestion, real estate prices and emissions).

All our use cases share a similar data flow where we can distinguish the same basic operations. We start from gathering data of different kind and origin. These data are described by other metadata which explain and complement their meaning. Next, data and metadata must be stored into a data management system which becomes a source of input for the processing systems. Relations between the data are described by a Domain Specific Language (DSL), which next builds a synthetic population and action patterns for the modelling systems. After running the simulations, the output data are sent to visualisation systems and presented in human-friendly way. A portal is used as a user interface for controlling and monitoring the whole process. Based on this general scenario we can develop a framework to generate customized synthetic populations for GSS applications.

This short description presents the complexity of the whole system. There are many parts which combined together should provide the desired functionality and efficient data processing.

Due to the global nature of our problem domains we can expect the analysis of huge datasets which on the one hand needs sophisticated data management systems and, on the other hand, significant computational power (HPC) to process this data efficiently. After running the simulations CoeGSS should provide its customers with detailed analyses, including real-time assessments, of global risks and opportunities.

The work on the presented scenarios (pilots) is the task of work package 4. A detailed description of the pilots is contained in deliverable D4.1. In WP3 we are taking requirements of the pilots and use them as drivers for realization of services which are then embedded into the service portfolio offered e.g. via the portal implemented in WP5.

The main goal of this deliverable is to present the state-of the-art and an evaluation of available methods, tools and mechanisms which can be integrated off-the-shelf within CoeGSS, to act as a baseline for future developments.

In chapter 3 we discuss tools which can be used for both efficient data management and maintenance of fault tolerance. Chapter 4 is devoted to remote and immersive visualisation systems potentially used within the project to explore, investigate and analyse GSS related data. Data types and data structures used within typical applications as well as addressed visualisation methods and practices will be summarized to identify an evaluation matrix. Chapter 5 presents a brief evaluation of available software for creating and running GSS synthetic populations, and for designing and implementing domain-specific languages (DSLs). Chapter 6 describes the monadic representation of different kinds of uncertainty and gives a short overview of existing implementations. Chapter 7 will first of all give a short introduction to the co-design approach in contrast to the classical design approach. Chapter 8 gives a short summary of the content.

2. Data management and fault tolerance

2.1 Introduction

GSS domains such as urban futures, green growth and global health are centrally data driven today with data generated in extremely high volumes. Data analytics are applicable to all such areas of science, engineering and innovation: they enable researchers to understand nuanced predictions, as well as shape policies and strategies more efficient and impactful. "Big data," machine learning, and predictive data analytics have been hailed as the fourth paradigm of science, allowing researchers to extract insights from both real world data and computational simulations. Machine learning has yielded new insights into health risks and the spread of disease via analysis of social networks, Web-search queries, and hospital data. It is also key to event identification and correlation in domains as diverse as energy modelling, human health and urban futures.

As we already mentioned, processing of GSS scenarios is related to computation of very huge sets of data. Moreover, we can expect a tremendous growth of the amount of data as well as their heterogeneity – taking into account different data sources, e.g. data streams from social media. It drives us to two main problems which should be solved in the proposed GSS framework: efficient data management and reliable computation.

Raw data in GSS are related with accompanying data (metadata) which provides more information about the meaning and validity of the provided numbers. That is why the process of gathering and storing of source information should include schemes which take into consideration all aspects of data. This schemes must facilitate the process of exploring data like: searching, comparing, combining, duplicating etc.

Data management systems used for GSS scenarios should have the functionality to store metadata according assumed schemes and the ability to provide huge numbers of data for processing without disrupting of (fast enough) computations. Computations (even within one scenario) can be executed on work nodes placed in different locations. In this situation a data management system should ensure that all necessary data will be located in the nearest location. Another situation when efficient data movement is crucial is situation when for some reasons (e.g. system crash) computations need to be relocated to another physical localisation. All input data should be moved to the best possible location right behind rerun tasks.

The High Performance Data Analysis (HPDA) services are designed for an emerging market (e.g. financial services, manufacturing, on-line retailers, telecommunications and healthcare/bioinformatics, energy) where simulation-based and analytics-based data analysis are complex enough to require the use of High Performance Computing (HPC) methods and resources.

Big Data describes a new generation of technologies and architectures designed to economically extract value from vary large volumes of a wide variety of data by enabling high velocity capture, discovery and analysis.

BIG DATA = Volume + Velocity + Variety

Crashes of computational systems happen with unspecified frequency and it must be assumed that they will also occur in our situation. Growing power of contemporary HPC clusters and their complexity causes a reduction of Mean Time Between Failures (MTBF). If response time of our services is important (usually is) we need to assure that in the case of failure a part or the whole of the computation could be completed on an alternative resource. As data analytics becomes more mission critical besides HPDA services also fault tolerance methodologies are becoming increasingly significant. In order to avoid very negative effects of these situations like the necessity of restarting a whole computation process from the beginning, waste of time and energy, the GSS processing should assume different scenarios of emergency circumstances.

Therefore, we can consider the main requirements for enabling HPDA are:

- Data movement appliance, geared towards end users, via NFS & CIFS
- Minimize admin responsibility
- Scalable, Highly Available, Load - balanced
- Proactive data management
- Ease to use - GUI-driven
- Enables automated end-to-end workflows (CLI)
- Role-based authentication (NIS & LDAP)
- Data mining, Audit history, etc.
- Export high performance file system e.g. Lustre

Below we discuss methods, techniques and implementations for efficient [1] data storage and movement as well as for fault tolerance management.

2.2 Data movement

It is assumed that about 90% of currently available data was generated in the last two years [2]. The number of data, which systems need to tackle with, is growing exponentially, it also relates to GSS systems. HPC data management systems manipulate very large amount of data by storing it on different levels of hierarchical storage (scratch disk, array disks, tapes).

Optimization of access to local file system can be achieved by optimization of I/O calls. In many cases it is made by parallelization of disk access operations. It is especially effective in situations when multiple processes try to read the same space on disk.

Another method of optimization is prediction. The system tries to predict which data will be required in the near future and load it into the memory in advance. Sometimes serval different scenarios are realized in parallel, that increases the probability that one will come true. This solution is often used by tape storage systems where time access to data is the longest.

2.2.1 Transmission protocol

The access to a big amount of data needs an efficient transmission protocol. Currently most of the tools responsible for data movement rely on TCP (based on sockets), whose flow is limited to 20Gbps. One way out is to use RDMA over Converged Ethernet (RoCE) [3]. This new network protocol used in high performed data movement allows remote direct memory access (RDMA) over an Ethernet network and reduces the CPU load. RoCE based solutions allows to use effectively several dozens of Gbps bandwidth using less CPU power.

2.2.2 Hierarchical Storage System

It is a common solution to use meta data in GSS systems in combination with a Hierarchical Storage System (HSS). In order to do it effectively we should access properly what level of meta data is used: performance, program or application.

In performance level I/O calls and prediction of data usage methods are used. Program level allows to implement easy to use interfaces for data access. This solution needs a description of the data structures which are used in applications, the location of the files, the offset of records within files. Finally, in the application approach programs are described by additional information like: time of run and arguments, execution environment, timing and results summary. All this information is used to elaborate application running pattern and predict which data could be needed. In order to find this pattern, we can use different solutions, the most popular is based on hints provided by the user and historical logs. In the first option the user may determine if data will be accessed by in temporal or spatial contiguity. By analysing historical logs the system can build patterns of data access which contain information about array of other files which with high probability can be accessed right after a given file.

2.3 Data management solutions

2.3.1 Hadoop

The Apache Hadoop [4] software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. Hadoop system was designed to automatically handle hardware failures. It allows storing distributed data sets with particular emphasis on computer clusters (scale up to hundreds or thousands of machines). Hadoop deals very well in situations when no data relationships exist and they are diverse in the nature. It contains the following main modules: Hadoop Common, Hadoop Distributed File System, Hadoop YARN, Hadoop MapReduce.

The Lustre [5] is a parallel file system that facilitates HPC simulation environments in many ways. Lustre is 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, harness the HPC power on very fast storage. However, there are also some disadvantages. The first of them is the overhead generated by HTTP calls during requiring data access by 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 meanwhile computations of the MapReduce.

2.3.1.1 MapReduce

The **MapReduce** programming paradigm breaks processing into two basic phases: a map phase and a reduce phase. The input and output of each phase are key-value pairs. The work is distributed across a set of parallel processors and executes smaller queries (mappers), then merging (reducing) phase coming that get data back together to return the full dataset.

2.3.1.2 Tez

Tez[1] is an application framework which allows for a complex directed-acyclic-graph of tasks for processing data and is built atop Apache Hadoop YARN.

2.3.1.3 Pig ETL abstraction

Pig ETL abstraction[2] - ETL jobs are written in a Pig-specific workflow language called Pig Latin, get compiled for execution by an underlying execution engine such as MapReduce.

2.3.1.4 Apache SAMOA

Apache SAMOA[3] is a distributed streaming machine learning (ML) framework that contains a programing abstraction for distributed streaming ML algorithms. Apache SAMOA enables the development of new ML algorithms without directly dealing with the complexity of the underlying distributed stream processing engines (DSPEe, such as Apache Storm, Apache S4, and Apache Samza).

2.3.1.5 Giraph

Giraph[4] is an open source implementation of Google's Pregel. From the ground up, Giraph is built for graph processing. Apache Giraph is an iterative graph processing system built for high scalability.

2.3.1.6 Storm Stream

Storm Stream[5] processing Storm has many use cases: real-time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, However, a repartitioning the streams between each stage of the computation is needed.

2.3.1.7 Apache Spark

Can replace MapReduce, Spark[6] offers SQL, graph processing (GraphX), machine learning algorithms implementation (MLib) and streaming frameworks. It provides a

[1] http://tez.apache.org/
[2] https://pig.apache.org/
[3] http://samoa.incubator.apache.org/
[4] http://giraph.apache.org/
[5] http://storm.apache.org/
[6] http://spark.apache.org/

simple programming model for varied range of applications and allows execute programs dozen times faster than Hadoop MapReduce. Developers are able to create complicated data flows e.g. directed acyclic graph. Spark supports sharing of data which are located in memory so in this way parallel tasks can operate on the same data set.

2.3.1.8 Flink

The general engine Apache **Flink**[7] is an open source platform for distributed stream and batch data processing. Flink offers Table API, graph processing (Gelly), machine learning algorithms implementation (FlinkML). It has dedicated support for iterative computations.

2.3.2 MongoDB

MongoDB[8] stores data as documents in a binary representation called BSON (Binary JSON). Documents that share a similar structure are typically organized as collections.

2.3.3 HBase

HBase[9] - it is a distributed, versioned, non-relational database modelled after Google's Bigtable.

2.3.4 Apache Cassandra

Apache Cassandra[10], a top level Apache project born at Facebook and built on Amazon's Dynamo and Google's BigTable, is a distributed database for managing large amounts of structured data across many commodity servers.

2.3.5 Amazon DynamoDB

Amazon DynamoDB[11] is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models.

[7] https://flink.apache.org/
[8] https://www.mongodb.com
[9] https://hbase.apache.org/
[10] http://cassandra.apache.org/
[11] https://aws.amazon.com/dynamodb/

2.3.6 Neo4j

Neo4j[12] is a graph database implemented in Java and accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint. The developers describe Neo4j as an ACID-compliant transactional database with native graph storage and processing.

2.3.7 S2Graph

S2Graph[13] is a distributed and scalable OLTP graph database built on Apache HBase to support fast traversal of extremely large graphs.

2.3.8 Apache Flume

Apache Flume[14] is a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store. The use of Apache Flume is not only restricted to log data aggregation. Since data sources are customizable, Flume can be used to transport massive quantities of event data including but not limited to network traffic data, social-media-generated data, email messages and pretty much any data source possible. The use of Flume within VElaSSCo will focus on providing the required pipelines to process simulation created data, in order to split it efficiently in several pieces that will be eventually stored in a NoSQL database, in a predefined schema. To do so, one of the requirements on VELaSSCo is creating one or multiple agents with Flume to synchronize on HBase repository. To do so, the pipelines are provided after multiplexing channels for agent. This is the case when we want to use the same agent over different channels to communicate between sources and sinks like in Figure below.

[12] http://neo4j.com/
[13] https://steamshon.gitbooks.io/s2graph-book/content/
[14] https://flume.apache.org/

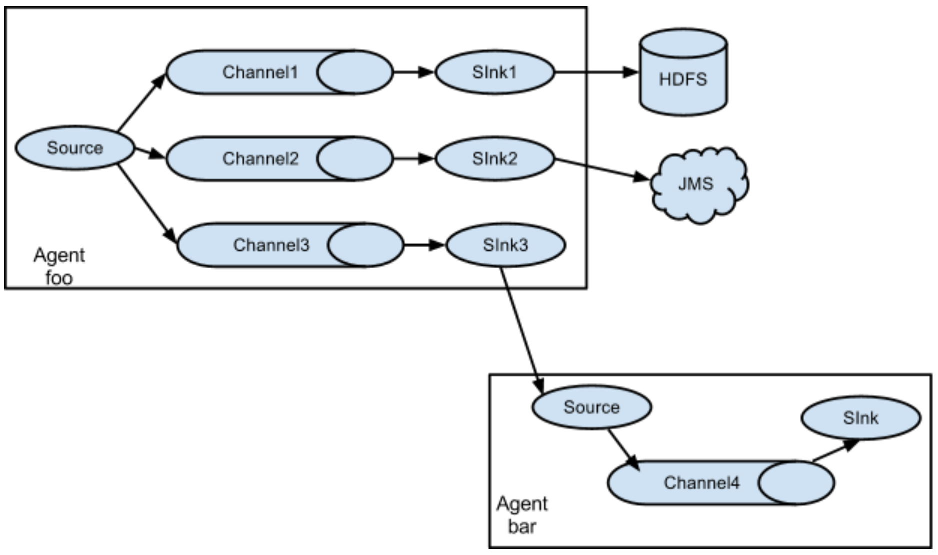

Figure 1. Using the same agent over different channels to communicate sources

Therefore, the Event Manager used is Flume, so the pipelines are created multiplexing different channels for the same agent, assigning different ports for each pipeline. The figure below represents the pipelines within the event manager.

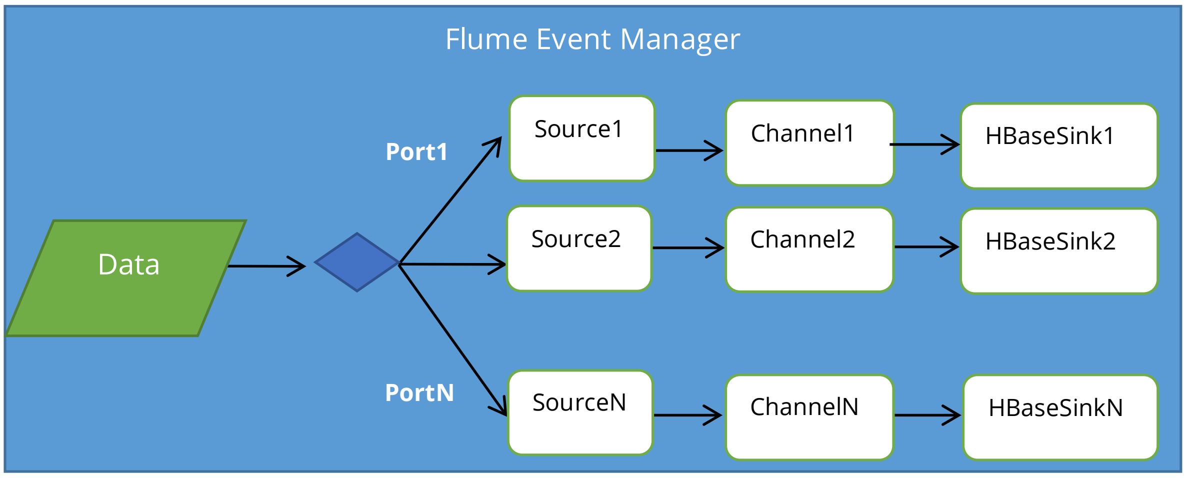

Figure 2. Flume Event Manager pipelines

2.3.9 NoSQL databases

NoSQL databases use other data models than the relational model known from the SQL world and do not necessarily adhere to transactional ACID properties. These models often sacrifice consistency and offer low level, non-standardized query interfaces that makes them harder to integrate in existing applications that expect an

SQL interface. However, these models provide a very good and efficient way of representing and storing data suitable for big data applications.

2.3.9.1 Key-value stores

Key-value stores allow the storage of data in a schema-less way. Data objects can be completely unstructured or structured and are accessed by a single key. As no schema is used it is not even necessary that data objects share the same structure. Examples of key-value stores are CouchDB[15], Oracle NoSQL Database[16], Dynamo[17], FoundationDB[18], HyperDex[19], MemcacheDB[20], Redis[21], Riak[22], FairCom c-treeACE[23], Aerospike[24], OrientDB[25] or MUMPS[26].

2.3.9.2 Columnar stores

Sometimes columnar stores are also called Big Table clones referring to Google's implementation of a columnar store. Such databases are typically sparse, distributed and persistent multi-dimensional sorted map in which data is indexed by a triple of a row key, column key and a timestamp. Examples are HBase[27], Cassandra[28], Accumulo[29], Druid[30] or Vertica[31].

2.3.9.3 Document databases

In contrast to the values in a key-value store, documents are structured. However, there is no requirement for a common schema that all documents must adhere to comparable to records in relational database. Thus document databases are referred to storing semi-structured data. Common encodings include XML or JSON. Examples

[15] http://couchdb.apache.org/
[16] http://www.oracle.com/us/products/database/nosql/overview/index.html
[17] https://en.wikipedia.org/wiki/Dynamo_%28storage_system%29
[18] http://www.foundationdb.com/
[19] http://hyperdex.org/
[20] http://memcachedb.org/
[21] http://redis.io/
[22] http://basho.com/products/
[23] http://www.faircom.com/
[24] http://www.aerospike.com/
[25] http://orientdb.com/orientdb/
[26] http://www.faqs.org/faqs/m-technology-faq/
[27] https://hbase.apache.org/
[28] http://cassandra.apache.org/
[29] https://accumulo.apache.org/
[30] http://druid.io/
[31] https://www.vertica.com/

are MongoDB[32], Apache CouchDB, Couchbase[33], DocumentDB[34], HyperDex, Lotus Notes, MarkLogic[35], Clusterpoint[36], OrientDB or Qizx[37].

2.3.9.4 Graph databases

Graph databases store data in graph structures making them suitable for storing highly associative data such as social network graphs. Examples are Neo4J 38 , InfiniteGraph39, OrientDB or Stardog40. A particular flavour of graph databases are triple stores such as AllegroGraph41, Virtuoso42 or GraphDB43 that are specifically designed to store RDF triples and can be used for semantic reasoning.

2.4 HPC fault tolerance

GSS systems perform complex and lengthy calculations where the risk of crash of the computational system is significant. Fault tolerance (FT) systems should prevent from interfering with operations in case of hardware and operating system failures. This solution is primarily intended for high-availability or life-critical systems. Fault tolerance allows to prevent partial results and restart (from previously saved snapshot) computation later saving already consumed energy and time.

Fault tolerance is also important from energy consumption point of view. Today's HPC systems consume huge amounts of energy which is wasted in case of a system fail.

2.4.1 Fault tolerance techniques

There are several fault tolerance techniques [6] well known and used in HPC solutions. The most important are the ones described below.

2.4.1.1 Checkpointing

Saving the current status of the process is made by the 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

[32] https://www.mongodb.org/
[33] http://www.couchbase.com/
[34] https://azure.microsoft.com/en-us/services/documentdb/
[35] http://www.marklogic.com/
[36] https://www.clusterpoint.com/
[37] https://www.qualcomm.com/qizx
[38] http://neo4j.com/
[39] http://www.objectivity.com/products/infinitegraph/
[40] http://stardog.com/
[41] http://franz.com/agraph/allegrograph/
[42] http://semanticweb.org/wiki/Virtuoso
[43] http://ontotext.com/products/graphdb/

and process is executed from the recently saved state, once the system governs the fault.

2.4.1.2 Replication

For effective execution several replicas (copies) of tasks are created and run on different resources. There are several different types of replication schemes e.g.: Active Replication, Semi-Active Replication and Passive Replication. Exemplary tools which can be used for the implementation of task replication are: Amazon EC2, Hadoop, HA-Proxy.

2.4.1.3 Resource co-allocation

This technique allows to allocate resources for further executions of a task. The allocation is made automatically depending on the properties as workload, type of task, capacity, energy awareness, etc.

2.4.1.4 Job Migration

Migration is performed if a machine fails and further execution there is not possible. The task is migrated to a working machine using a HA-Proxy. Specialized algorithms automatically determine the fault and migrate batch applications within multiple datacentres.

2.4.1.5 Task Resubmission

In the situation when a task fail is detected, it is resubmitted to different working resources for re-execution and new work is assigned by a job distribution algorithm.

2.4.1.6 Timing Check

This technique keeps track of the task execution. Depending on whether the task has been completed in the required amount of time or not further action for fault tolerance is taken.

2.4.1.7 Rescue workflow

The execution of the whole process is divided into chunks and represented as a workflow. Each step is after another until it becomes impossible – moving forward needs repairing of the failed task.

2.4.1.8 User defined

In the case of fault, the action to be performed is defined by the user. When unexpected situations happen a special procedure is executed.

2.4.2 Fault tolerance implementations

Some of the most popular implementations of the fault tolerance systems are described below. Many other, not mentioned in this ranking, are problem-specific solutions designed for individual software and taking into consideration its specifics.

2.4.2.1 SWIFT

Software Implemented Fault Tolerance [7] (SWIFT), by integration with compilers, provides protection by control-flow checking mechanism. In this technic validation points are not needed any more what results in an increase of performance (ca. 14% comparing to singled-threaded approach). It exploits unused instruction-level parallelism resources present in the execution of most programs to efficiently manage fault detection.

2.4.2.2 Berkeley Lab Checkpoint/Restart

This implementation combines user and kernel level of checkpointing for a wide range of applications. Very interesting is that BLCR [8] does not require changes to be made into application code. The main focus is on parallel applications that communicate through MPI. Project covers the following areas: Checkpoint/Restart for Linux (CR), Checkpointable MPI Libraries, Resource Management Interface to Checkpoint/Restart, Development of Process Management Interfaces.

2.4.2.3 Fault Tolerance Interface

Fault Tolerance Interface (FTI) [9] is a library that aims to provide application-level checkpointing to computational scientists with convenient way to perform restart in a scalable fashion. In order to improve efficiency and avoid space, time and energy waste it allows to select data which are protected by direct data interface. It is also possible to assign one process per node to overlap scientific computation and fault tolerance workload, then post-checkpoint job is executed asynchronously.

2.4.2.4 Distributed MultiThreaded Checkpointing

Distributed MultiThreaded Checkpointing (DMTCP) [10] is another tool which does not require modification of the user's application. Checkpointing can be made on multiple machines for specified group of socket connected applications. DMTCP is supporting the following programming languages: Python, Perl and Open MPI. It supports the OFED API for InfiniBand on an experimental basis. DMTCP facilitates checkpointing of a wide range of Linux functionality like: signal handlers, process id, pipes, sockets, open file descriptors and many others.

3. Remote and immersive visualization systems

The following chapter will give a brief introduction to remote and immersive visualisation systems potentially used within the project to explore, investigate and analyse GSS related data. Data types and data structures as used within typical applications by the consortium as well as addressed visualisation methods and practices will be summarized to identify an evaluation matrix. This will allow a comparison of listed visualisation systems and identify further visualisation functionality, which has to be added or implemented within the project.

3.1 Introduction

GSS simulations provide huge amounts of complex data. A lot of work has previously done on the visualization of large 3D data coming from measurements as well as numerical simulations and there is a growing field of multidimensional and “Big Data”, mainly in 2D. However, there is still a major gap to fill: how can we immerse analysts in the data in order to provide more natural ways for data exploration, data analysis and collaboration?

Driven by the entertainment industry, tools and methods like voice- and gesture-based control, multi-touch devices, tiled displays as well as 3D VR and AR visualisation are becoming more and more part of everyday life. In response to these new technologies, we propose the development of “Immersive Analytics Environments”. The environments we envision should be usable by experts and analysts to help in the detailed analysis of complex data sets, but will also be accessible to decision makers and politicians who spend more time working face-to-face than in front of a desktop computer, and to the everyday public in order to better involve them in the decision making processes.

Not all display hardware is equally suited for all types of data analysis. Whereas CAVEs are most effective in the display of complex 3D data, large high resolution tiled displays are better suited for huge amounts of 2D data. Similar is true for multi-touch displays, tangible interfaces, touch tables, tablets or HMDs. All have their special application field. In our Environment, we want to combine a multitude of different visualization and interaction devices in one seamless working environment and still allow access to large scale HPC and Data Storage systems.

Within the project the focus is to provide remote and immersive visualization to the consortium partners, to develop remote visualization services in order to provide interactive access to HPC and Visualization resources, to integrate 2D and 3D visualization systems in a seamless manner in order to create “Immersive Analytics

Environments" as well as to develop immersive visualization methods for huge statistical and multidimensional datasets.

3.2 Requirements

The following paragraphs will give a survey of project relevant requirements as they could be defined by the consortium so far. It is obvious, that within the early stage of the project, various requirements are given sketchy. So criteria evaluating remote and immersive visualisation systems focus primarily on availability and expandability of relevant features and software modules. More general features of visualisation systems, like hardware requirements or rendering features for instance, are not listed.

General requirements

- The GSS synthetic population simulations will be the main tool used in CoeGSS, in particular, within the pilot projects (see 6.2). Used simulation systems should be capable to read these types of data or provide an interface to the software being used.
- 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 the simulation is running.

Visualisation methods

- The visualisation system should provide methods to process huge and varying volumes of unstructured data in GSS.
- The visualisation system should provide methods for additional data management.
- The visualisation system should provide or interface to high performance data analysis tools.

Data types

- Results from simulation and analytics
- Structured and unstructured data
- Regular and irregular patterns (lists, matrices, graphs)
- GSS Models
- Incomplete information
- Inventory and classification of validated numerical methods useful for generating synthetic populations, especially type theoretical approaches for interval based computations.

Requirements for rendering environments

- 3D virtual environments like CAVEs or a Powerwall
- 2D Desktop

Interfaces to specific tools

- Hadoop (see 3.3)

For classification and evaluation of relevant visualisation systems, we will use the following list of criteria:

- data processing capability

  The visualisation system should be capable to process the data in a way, so that data sets could be investigate from various viewing directions or in different arrangements for instance. This will support analysing and interpreting capabilities for the project.

- I/O modules / general expandability

  The visualisation system should be capable to interface external software systems and to import data sets from these systems. A common approach is to provide a set of reader or import modules like ReadCSV for instance, which interprets the output of the simulation systems and provides an assisted processing for visualisation.

- API

  The software system should provide an API, which enables the user 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.

- license / availability

  The software should be available to the consortium and currently being maintained.

- supported hardware environments

  The visualisation software should be capable to run on all common platforms (Microsoft Windows, Mac OS X, Linux). The visualisation software should support local 2D rendering and interaction on common workstation displays as well as 3D rendering and interaction for virtual reality environments like powerwalls and CAVEs [11].

- remote visualisation

The visualisation software should support remote rendering / visualisation, so that the visualisation software can be used on remote workstations or compute / rendering clusters as well as on high performance computing resources. It should provide capability to interface web based access to the visualisation.

3.3 Remote and immersive visualisation systems

The following paragraphs will give a survey of remote and immersive visualisation systems which are currently available and potentially relevant for the project.

3.3.1 The Chromium Project

Chromium [12] is a system for interactive rendering on clusters of graphics workstations. Various parallel rendering techniques such as sort-first and sort-last may be implemented with Chromium. Furthermore, Chromium allows filtering and manipulation of OpenGL command streams for non-invasive rendering algorithms.

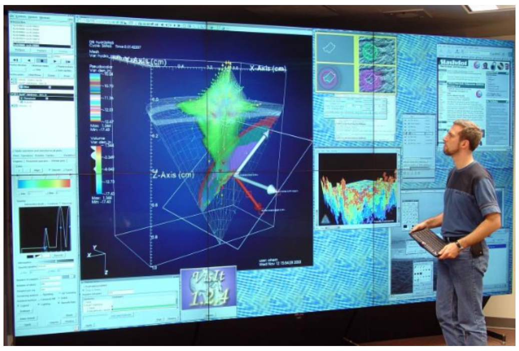

Figure 3. The Chromium Project

Functionality

Among Chromium's features are

- Sort-first (tiled) rendering - the frame buffer is subdivided into rectangular tiles which may be rendered in parallel by the hosts of a rendering cluster.
- Sort-last (Z-compositing) rendering - the 3D dataset is broken into N parts which are rendered in parallel by N processors. The resulting images are composited together according to their Z buffers to form the final image.
- Hybrid parallel rendering - sort-first and sort-last rendering may be combined into a hybrid configuration.
- OpenGL command stream filtering - OpenGL command streams may be intercepted and modified by a stream processing unit (SPU) to implement non-photorealistic rendering (NPR) effects, etc. Here's an interesting example.
- Many OpenGL programs can be used with Chromium without modification.

- One can write Chromium-specific applications which perform parallel rendering with the aid of special synchronization primitives.

Chromium is derived from the WireGL project[44].

Platform / Support

- Microsoft Windows, Linux, IRIX

Programming language / API

- Python, C

License

- BSD License
- The Chromium Project is available open source, but is no longer updated or maintained. The project is frozen since January 2015[45].

Organisation / Main Contributors

- Standford Computer Graphics Laboratory[46]

3.3.2 COVISE / OpenCOVER

COVISE [13] stands for COllaborative VIsualization and Simulation Environment. It is an extendable distributed software environment to integrate simulations, postprocessing and visualization functionalities in a seamless manner. From the beginning COVISE was designed for collaborative working allowing engineers and scientists to spread on a network infrastructure. In COVISE an application is divided into several processing steps, which are represented by COVISE modules. These modules, being implemented as separate processes, can be arbitrarily spread across different heterogeneous machine platforms. Major emphasis was put on the usage of high performance infrastructures such as parallel and vector computers and fast networks. COVISE Rendering modules support Virtual environments ranging from workbenches over powerwalls, curved screens up to full domes or CAVEs [14]. The users can thus analyse their datasets intuitively in a fully immersive environment through state of the art visualization techniques including Volume rendering and fast sphere rendering. Physical prototypes or experiments can be included into the analysis process through Augmented Reality techniques.

[44] http://graphics.stanford.edu/software/wiregl/
[45] http://sourceforge.net/projects/chromium/
[46] http://graphics.stanford.edu/

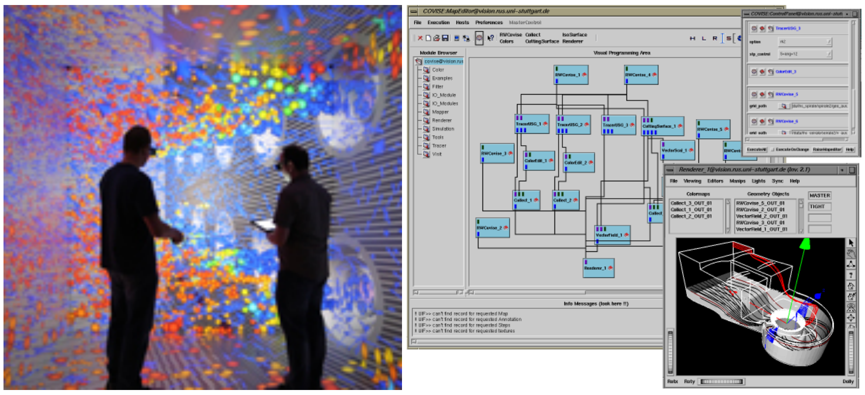

Figure 4. COVISE / OpenCOVER

Functionality

The window in the above figure shows the so called MapEditor which is the main part of the user interface. It is a Qt based program to allow interaction with COVISE. In a collaborative session there are as many user interfaces as there are participants. One user is the master and has the complete control over the environment while the others are slaves and can do nothing besides requesting the master role. The large area in the middle is for placing modules. Connected they form a flow chart like execution pipeline. By clicking on a module in the module list the user places the desired modules in the MapEditor window. Connecting the modules on a Point-and-Click basis defines the data flow between the modules.

The main processes are the User Interface (UI) called MapEditor, the Controller, the COVISE Request Broker (CRB) and the application modules. The local workstation normally is the one where COVISE is started. On this machine the user interface pops up and the Controller is started. All other processes are created from here, either using exec calls for local processes or rexec/rlogin/rsh/ssh for processes on remote computers. The user can include additional hosts into a session for remote module execution. On each machine, a shared data space exists, which normally consists of shared memory. The CRB administers the use of this shared data space in a database-like fashion.

COVISE allows several users to work in a collaborative way. One can invite other users on different hosts to participate in the current COVISE session. The user who initiated the collaborative session becomes the master. On all machines a user interface pops up and a CRB is started. Application modules can be started on any host participating in the session. Renderer modules play a special role: in a collaborative session Renderers run locally on each machine. When the master user manipulates the objects in the Render window, only small synchronisation information has to be sent to the other renderer modules. When the master rotates the scene, the new transformation matrix is sent to the controller which in turn sends it to all other (slave-) Renderers. Of course every user can request the master role. For audio- and videoconferencing standard tools like InPerson or the public domain MBONE tools VIC/VAT are usually used.

Own simulation or rendering codes can be integrated into COVISE as new modules by adding a few calls from the COVISE libraries to create COVISE objects from data and for communication with other COVISE modules.

The modular structure of COVISE makes it very easy to integrate new modules for special tasks. The complex three-dimensional structure of the datasets that are visualized with COVISE made us look for more advanced visualization techniques. The result was a virtual reality rendering module called COVER (COVISE Virtual Environment), which is based on Iris Performer. COVER has been replaced by OpenCOVER which is based on OpenSceneGraph.

OpenCOVER is a further development of COVER, which was the original COVISE VR renderer. It is based on OpenSceneGraph, an open source high performance 3D graphics toolkit, to make optimal use of the rendering hardware. OpenCOVER stands for "Open COVISE Virtual Environment" and is an integral part of the COVISE visualization and simulation environment. The design of OpenCOVER was developed to support technical and scientific applications. OpenCOVER supports loading VRML2.0/VRML97 datasets, Detailed information about the supported features can be found in this compatibility matrix.

The user interacts with the scene through the 3D device and a 3D menu. A line coming out of the device intersects with the menu and the scene. Depending on the selected action an appropriate interactor is drawn. The pictures on the left show some of the interactors. Pressing the buttons selects a menu item or object. The pictures on the left shows some of the interactors that can be attached to the stylus. Feedback is a very special kind of interaction: it's interaction with other COVISE modules like on-going numerical simulations and post-processing steps.

As a COVISE renderer module OpenCOVER also supports collaborative working. Three collaboration modes have been implemented, Loose Coupling, Tight Coupling, and Master/Slave Coupling. Cone-shaped markers can be placed in the 3D scene to point to positions of interest.

OpenCOVER was developed for virtual environments using back-projection screen based environments like CAVEs. The screen size can be configured easily through a configuration file.

Platform / Support

- UNIX/Linux, Windows, MacOS

Programming language / API

- C++, VPL (Map-Editor)

License

- LGPL v2.1
- COVISE and OpenCOVER are available open source[47].

Organisation / Main Contributors

- High Performance Computing Center Stuttgart (HLRS)
- University of Cologne, Chair of Computer Science

3.3.3 Ensight

EnSight is a software program for visualizing, analysing, and communicating data from computer simulations and/or experiments. Major fields of application for EnSight are Automotive, Aerospace, Defense, Combustion, Energy Production, High-Tech Manufacturing, and other fields which require very high precision in computer-based physics modelling. It is most often used for Computational Fluid Dynamics (CFD), Computational Structural Mechanics (CSM), and other CAE (computer aided engineering) processes.

EnSight is a neutral post-processing tool with interfaces to the most popular CFD packages such as Fluent, Star-CCM+, Converge, CFX, Open-FOAM etc., and popular FEA interfaces such as Ansys, Abaqus, Nastran, LS-DYNA etc. EnSight also offers CAD interfaces for all the popular CAD systems, such as Catia V4 and V5, Pro/Engineer etc. In addition to this, there are interfaces for IGES, STEP for instance.

[47] https://github.com/hlrs-vis/covise

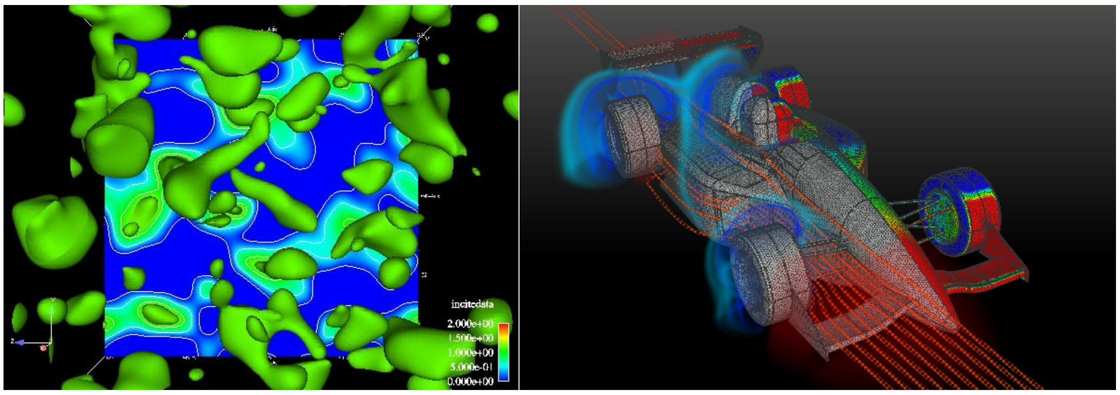

Figure 5. CEI EnSight[48]

Functionality

As described in [15], the vision behind EnSight has been a full-featured, interactive, high performance visualization tool capable of scaling to the largest data sets. At the centre of that vision is the ability to effectively leverage advanced computer systems both at the desktop/display and in computational clusters. This chapter reviews the EnSight framework for distributed application launch and rendering. The distributed launch system facilitates custom deployment of scalable EnSight visualization solutions. It has evolved to meet the increasingly varied requirements of its users, including heterogeneous environments with potentially complex and restrictive access controls. The rendering system is specially geared toward improved user interaction and advanced rendering in distributed, high performance scenarios by exploiting advances in desktop graphics card technologies and direct interaction methodologies.

CEI has kept an eye on developments in virtual reality (VR). As 3D tracking devices, input devices, and new display technologies matured, CEI integrates VR capabilities into its family of products [16].

This article [17] is a look at VR and how it is being implemented to help EnSight users better analyse, visualize and communicate engineering and scientific results.

Platform / Support

- Windows, Linux/Unix, Max

Programming language / API

- build-in Python scripting

License

[48] http://vis.lbl.gov/NERSC/Software/ensight/

- commercial

Organisation / Main Contributors

- Computational Engineering International, Inc. (CEI Inc.)[49]

3.3.4 ParaView

ParaView [18] is an extensible, reconfigurable framework that is used to inspect data in many forms. The entire project is open source with a very liberal license and has been designed from the start to be reused and extended wherever its capabilities are useful. The desktop application platform is just one configuration of the modular, parallel components that make up ParaView. Other “flavors” include the extensible and reproducible python interface, immersion in advanced display environments, in situ coupling with simulation code via Catalyst, ubiquitous access through web based visualization and of course scalable processing in High Performance Supercomputing (HPC) platforms.

ParaView [19] was developed to analyse extremely large datasets using distributed memory computing resources. It can be run on supercomputers to analyse datasets of petascale size as well as on laptops for smaller data, has become an integral tool in many national laboratories, universities and industry, and has won several awards related to high performance computation.

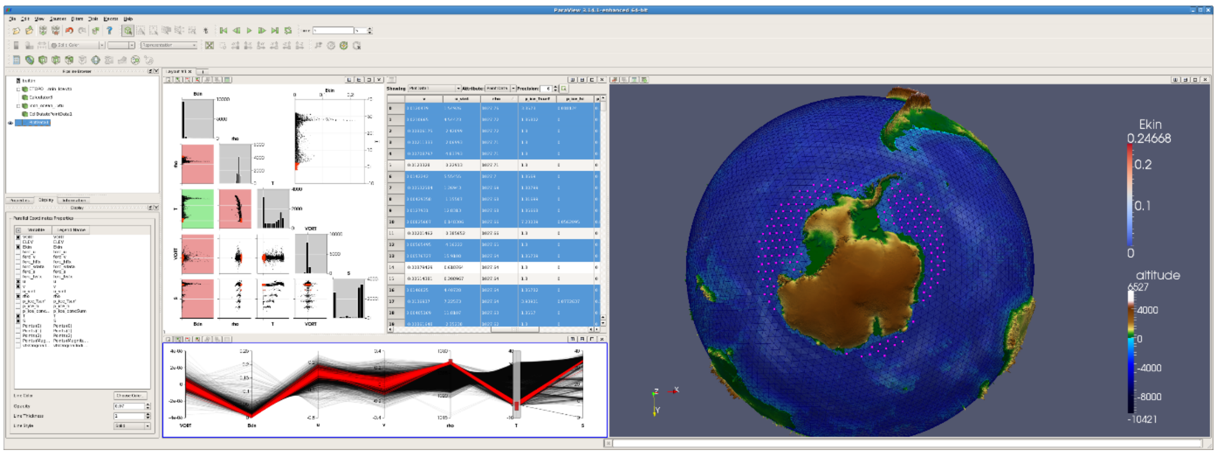

Figure 6. ParaView[50,51]

[49] http://www.ceisoftware.com/
[50] http://www.paraview.org
[51] https://www.dkrz.de

Functionality

The most frequently used interface to ParaView is the desktop application. This is a Qt-based application for Mac, Windows and Linux operating systems in which users very easily open data files and begin to visualize them.

Seamless integration with Python began in ParaView version 3.0. By simply loading a module from Python the user gets full access to all of ParaView's large data visualization and analysis capabilities. This includes the ability to create, on the fly, scripted readers and filters that run, in parallel, on the server. ParaView scripts are easy to write, especially if you choose to simply record your work in the desktop application in the form of a python script. Python scripts can be played back with or without the GUI in order to create reproducible, easily customizable, and scalable visualizations.

It is already common for simulations to discard most of what they compute in order to minimize time spent on I/O. As we enter the exascale age the problem of scarce I/O capability continues to grow. Since storing data is no longer viable for many simulation applications, data analysis and visualization must now be performed *in situ* with the simulation to ensure that it is running smoothly and to fully understand the results that the simulation produces. Catalyst is a light-weight version of the ParaView server library that is designed to be directly embedded into parallel simulation codes to perform *in situ* analysis at run time [20].

ParaView's stereo display modes, efficient tile and cave rendering engine, and VRPN tracking interfaces make such immersion in the data possible.

Using the aggregate disk space, processing power and especially memory of a cluster or supercomputer, ParaView scales to allow visualization and analysis of even the largest scientific results. The pvbatch and pvserver family of executables are MPI enabled programs that run in parallel on distributed memory parallel computers, in which each node in the machine processes only a small portion of the entire data. The many results are finally combined together and rendered, either in parallel when the resulting geometry is large or in serial when the geometry is small, and displayed.

Web applications can embed interactive 3D visualization components via ParaViewWeb's light-weight JavaScript API. Underneath this API, ParaVewWeb communicates with and responds to a ParaView server instance running on a remote visualization node or cluster through HTML 5.0 based technologies, such as WebSockets and WebGL. Together you have the accessibility of the world wide web combined with the scalability of the ParaView server.

Platform / Support

- Unix/Linux, Mac OS X, Windows

Programming language / API

- C/C++, Fortran, Python

License

- BSD

Organisation / Main Contributors

- Sandia National Laboratory,
- Kitware Inc.
- Los Alamos National Laboratory

3.3.5 SCIRun

SCIRun [21] is a problem solving environment or "computational workbench" in which a user selects software modules that can be connected in a visual programing environment to create a high level workflow for experimentation. Each module exposes all the available parameters necessary for scientists to adjust the outcome of their simulation or visualization. The networks in SCIRun are flexible enough to enable duplication of networks and creation of new modules.

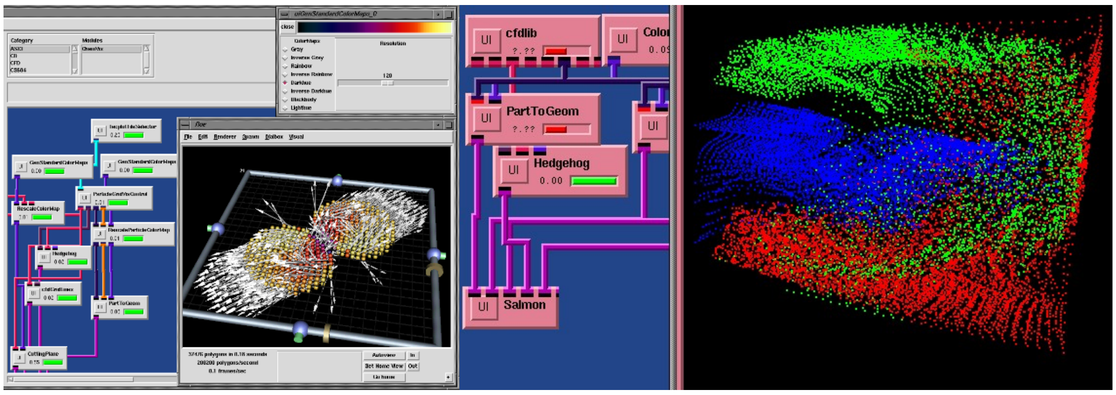

Figure 7. SCIRun[52,53]

Functionality

The goal of integrated problem solving environments like SCIRun [22] is to integrate data processing as components in a single, unified, extensible problem solving

[52] http://www.vacet.org/
[53] https://www.sci.utah.edu

environment. The resulting function includes the ability to manage each step in a sequential computing process, and to create batch processes that execute repeated simulations. The functionality that sets SCIRun apart from most integrated software environments is the ability to intervene and control execution anywhere in the chain at any time during its execution. The ability to control a computer program during execution is termed computational steering.

In the example of the defibrillation simulation, computational steering allows users to interactively change parameters and settings as the simulation executes, performing his or her work in batch and interactive modes. Steering interventions might include adjusting electrode locations to stay within anatomically reasonable bounds, or refining the geometric model resolution in order to balance accuracy and execution time.

To achieve integration within the elements of SCIRun, data flows directly from one processing step to the next, without being diverted to a disk file or leaving the program. Output from each step is available as input to dependent steps. The underlying paradigm of SCIRun is data flowing between modules that each perform some operation. Integration between modules guarantees, that upon completion of their tasks, upstream modules pass their data to downstream modules, thereby forcing the downstream modules to execute in response. In the computational steering example, the user may alter electrode locations at any time, initiating a sequence of all necessary steps to recompute the simulation with the new configuration. The modification of the geometric model, finite element calculation, and visualization all proceed automatically and in the proper sequence, all managed by SCIRun. The combination of steering and component integration allows the user to spontaneously explore a problem.

While computational steering is a young field in computer science, there are a number of examples of such systems (in addition to SCIRun) described in the literature. SCIRun can be extended by creating new packages, modules, and subnets. Modules can be coded from scratch, or with the assistance of the SCIRun's Module Maker component. SCI encourages users to contribute modules to the BioPSE web site CIBC web site where they will be reviewed, and useful modules will be included in subsequent releases of SCIRun. To leverage investment in legacy code, future releases of SCIRun will include additional tools for wrapping existing code within SCIRun modules.

SCI is developing problem specific applications called Power Apps. Power Apps use SCIRun's data flow engine for computation and application specific data flow networks for problem solving. However, Power Apps hide the complexity of the data flow environment behind a simplified application specific graphical user interface [23].

Platform / Support

- Windows, Max OS X, Linux

Programming language / API

- C++, Python

License

- The MIT License

Organisation / Main Contributors

- University of Utah, Scientific Computing and Imaging Institute[54]

3.3.6 ViSTA VR toolkit

The ViSTA virtual reality toolkit aims to enhance scientific applications with methods and techniques of Virtual Reality and immersive visualization, thus enabling researchers from multiple disciplines to interactively analyse and explore their data in virtual environments. ViSTA is a software platform that allows integration of VR technology and interactive, 3D visualization into technical and scientific applications. ViSTA FlowLib [24] combines rendering techniques for the exploration of unsteady flows in virtual environments.

The toolkit [25] has been developed at the Virtual Reality Group at RWTH Aachen University for more than ten years and, recently, a cooperation in its development with the DLR Braunschweig was established.

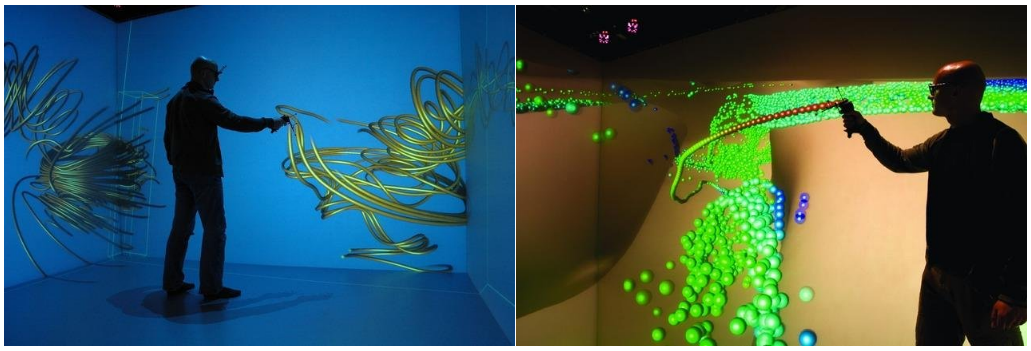

Figure 8. ViSTA VR toolkit

[54] https://www.sci.utah.edu/software/scirun.html

Functionality

The VistaCoreLibs provide basic and commonly use functionality for virtual reality application, such as

- Display Management
- Device input for a variety of VR input and output devices, e.g. tracking cameras or haptic devices.
- Input device abstraction using a dataflow network
- Coupling to the OpenSG scenegraph library (OpenSceneGraph - support planned)
- Cluster mode for distributed execution of ViSTA applications for multi-display environments
- Utilities for system-independent use of threads, connections, files, timers, etc.

The VistaFlowLib provides methods for scientific visualization

- Data and time management
- Interaction methods
- Rendering methods
- Particle tracing

The VistaAddonLibs provide specialized functionality, e.g.

- Collision and physics simulation
- Methods for medical simulation (e.g. Haptics, Softbody simulation)
- Audio interface

Platform / Support

- linux x86, linux x86-64, win32, win32-x64

Programming language / API

- C++

License

- GNU Library or Lesser General Public License version 3.0 (LGPLv3)
- The ViSTA core libraries are available open source[55].

[55] http://sourceforge.net/projects/vistavrtoolkit/

Organisation / Main Contributors

- RWTH Aachen - Virtual Reality Group (VRG)[56]
- DLR - Software for Space Systems and Interactive Visualization[57]

3.3.7 The ViSUS Visualization Framework

The ViSUS software framework [26] has been designed as an environment that allows the interactive exploration of massive scientific models on a variety of hardware. Furthermore, it has the ability to do so over platforms distributed geographically.

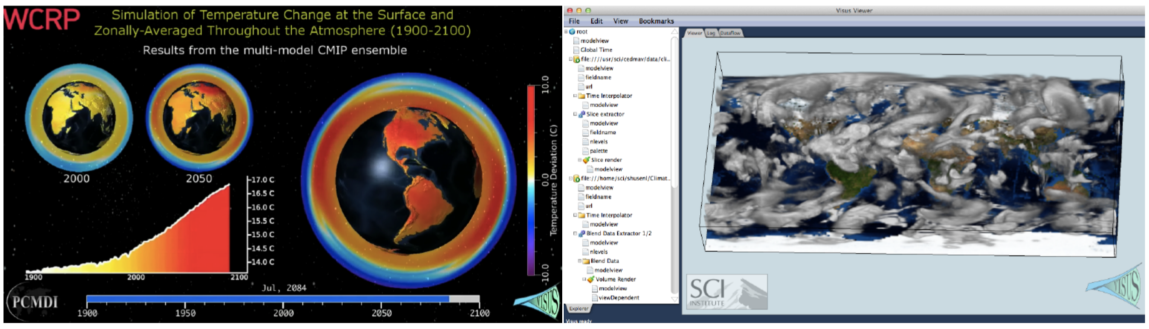
WCRP
Simulation of Temperature Change at the Surface and Zonally-Averaged Throughout the Atmosphere (1900-2100)
Results from the multi-model CMIP ensemble
2000
2050
Temperature Deviation (C)
Jul, 2084
PCMDI
ViSUS
SCI INSTITUTE

Figure 9. The ViSUS Visualization Framework[58]

Functionality

The ViSUS software framework was designed with the primary philosophy that the visualization of massive data need not be tied to specialized hardware or infrastructure. In other words, a visualization environment for large data can be designed to be lightweight, highly scalable, and run on a variety of platforms or hardware. Moreover, if designed generally, such an infrastructure can have a wide variety of applications, all from the same code base. The components can be grouped into three major categories: First, a lightweight and fast out-of-core data management framework using multiresolution space-filling curves. This allows the organization of information in an order that exploits the cache hierarchies of any modern data storage architectures. Second, a data flow framework that allows data to be processed during movement. Processing massive data sets in their entirety would be a long and expensive operation, which hinders interactive exploration. By designing new algorithms to fit within this framework, data can be processed as it moves. The third category is a portable visualization layer, which was designed to scale from mobile devices to Powerwall displays with same the code base. This chapter describes the ViSUS infrastructure, and also explores practical examples in real-world applications.

[56] https://www.itc.rwth-aachen.de/cms/IT-Center/Forschung-Projekte/~eubl/Virtuelle-Realitaet/
[57] http://www.dlr.de/sc/en/desktopdefault.aspx/tabid-1200/1659_read-3101/
[58] http://visus.net/

Platform / Support

- Mac OS, Linux, Windows

Programming language / API

- C++

License

- commercial

Organisation / Main Contributors

- ViSUS LLC, Salt Lake City[59]

[59] http://visus.net

3.4 Comparison

The relevant software systems (see 4.3) are evaluated according to the criteria (see 4.2).

	Chromium	COVISE/ OpenCOVER	Ensight	ParaView	SCIRun	ViSTA	ViSUS
data processing	o	++	o	++	++	++	++
I/O modules	o	++	o	++	++	++	++
API	++	++	-	++	++	++	+ [2]
license	o	++	-	++	++	++	-
hardware support	+	++	+	++	++	+	+
remote visualisation	++	++ [1]	+	++	++	++	++

Table 1. Visualisation software evaluation

[1] currently in development, [2] API was not available for testing

4. Methods, tools and mechanisms for GSS and building DSLs

4.1 Introduction

Synthetic information systems (SIS) will be the main tool used in CoeGSS, in particular, within the pilot projects.

Tackling a problem with an SIS involves several steps, roughly divided into five phases:

1. Determining the static aspects of the SIS:

   This phase involves

   – designing the (human) agents: selecting the features that will be represented (such as age, income, employment status, health, etc.)
   – designing the environment of the agents: selecting relevant locations (schools, workplaces, homes, etc.), other objects (cars, houses, communication technology, etc.), resources (water infrastructure, gas stations, etc.), agencies not modelled directly as collections of agents (e.g., international law-giving bodies, government measures, etc.)
   – establishing statical relationships between the various elements (kinship, ownership, subordination, etc.)

   If we look at the SIS as a dynamical system, this phase contributes towards the definition of the state space of the system. The state is viewed as a structured collection of individual states of the agents and components of the environment.

2. Determining the dynamical aspects of the simulation:

   – choosing the activity patterns of the agents, usually in the form of timetables (agents that usually go to work on week-days, or to school, or work shifts, etc.)
   – establishing the dynamical relationships between the elements (the probability that an agent will visit a given location on a given day at a given time, the probability that the government will issue a certain measure depending on the state of the environment, etc.)
   – selecting the dynamics of the simulation itself (deciding whether locations can be added during the running of a simulation, which changes can be made to the parameters, allowing for checkpointing, allowing for user interaction, etc.)

   We underline that the static and dynamic aspects of the SIS are not independent of each other. For example, the choice of activity patterns will often lead to changes in the features that must be represented in the agents.

   This phase determines the type of the transition function of the system, usually seen as composed of "local" transition functions of the agents, environment, and of the simulation infrastructure. This is not yet a *definition* of the transition function, since most of the parameters will only be fixed in the next phase.

3. Obtaining and processing the data and creating the SIS
   - discovering the relevant data sources, downloading data, "scrubbing" data, etc.
   - creating a synthetic population such that the statistical properties of its agents match those of the real intended population in the case of the selected features (as given by the data)
   - identifying the parameters of the local state transition functions (the probability that agents will visit certain locations under given circumstances, weather patterns, etc.)

   The distinction between static and dynamical aspects of the SIS reflects the fact that data for creating the static aspects is usually more readily available (as, e.g., census data) than the data for transition probabilities.

   In this phase, the initial state and the transition function for the SIS are fully determined: the simulations can be run.
4. Running the SIS simulations
5. Analysing and interpreting the resulting data

In many cases, interactions between agents are mediated by locations (agents interact by virtue of being in the same place at the same time). The transition functions of agents are then represented using *activity patterns* as building blocks:

Activity	Monday	Tuesday	...
morning	Work(0.7), Home(0.3)	...	
afternoon	Work(0.6),Home(0.3),Gym(0.1)	...	
evening	Home(0.7),Restaurant(0.2),Movie(0.1)	...	
night	Home(0.9),Hotel(0.1)	...	

Table 2. Activity patterns as building blocks

The entries in the table represent probability distributions over the possible locations the agent will find itself at the given time. For example, if the agent is a schoolchild, there will be a high probability that it finds itself in school during school day mornings, the same holds true for school teachers.

Except for the simplest cases, an agent has several activity tables associated with it, the one being used will depend on its state and the state of its neighbourhood. For example, sick schoolchildren are much likelier to be at home than in school, even on school days. Healthy schoolchildren with sick parents are also likely to have their schedules disturbed, etc. In general, the more complex the agents and their interactions are, the more information has to be supplied outside of the activity tables in order to compute the transition function. Nevertheless, the concept of activity pattern plays an important role in many applications and is a key element in the creation of DSLs for GSS synthetic information systems. Tabular representations of functions are common DSL patterns [27] and are good starting points for graphical user interfaces geared towards non-programmers.

Widening the circle of potential users of GSS SISs is only a secondary benefit of the DSL approach. Its main contribution consists in improving the understanding of the concepts involved ("agent", "population", etc.), being, in effect, a formalisation of these concepts, and in providing a uniform access to the software packages that implement the creation of the synthetic population and carry out the simulation.

As an example, consider the open source framework for epidemic modelling FRED [28]. In FRED, the characteristics of the agents are hard-coded, as are many aspects of the transition functions, all being focused on the goal of efficient simulation of epidemics in populations with tens or hundreds of millions of agents. In order to use FRED for a different purpose, such as modelling the spread of health-relevant habits in a population, one has to modify virtually every single file of the framework code. Nevertheless, the general structure and most of the code are applicable to such purposes as well. A DSL for GSS SISs would 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 FRED framework, in much the same way as a compiler of a high-level language such as Eiffel can generate code written in C. Just as the reason for compiling code to C is to take advantage of the existing environment for C programs, so the reason for compiling a GSS SIS to FRED is to take advantage of the FRED framework. If a different framework turns out to be more suitable, the GSS SIS does not need to be changed: a new back-end can be provided, which uses the new framework instead of FRED. By separating the conceptual part from the implementation the DSL approach avoids lock-in and enables us to take advantage of resources as needed.

The next section presents a number of existing software packages that can be used as back-ends for the DSLs for GSS SISs. The final section is dedicated to methods and tools for implementing the DSLs themselves.

4.2 Available software for creating and running synthetic populations

4.2.1 Software for creating synthetic populations

4.2.1.1 simPop

- *Description*: simPop is an Open Source R Package for Generating Synthetic Populations

- *Functionality*: The package provides a way of generating synthetic populations of households.
- *Programming Language*: R programming language
- *License*: GPL v2/3
- *URI*: http://www.ihsn.org/home/projects/synthetic-populations
- *Organisation*: International Household Survey Network (IHSN)

4.2.1.2 PopGen

- *Description*: Tool for synthesizing populations in small geographies, used for activity based modeling
- *Functionality*: Generates synthetic populations while controlling and matching both household-level and person-level attribute distributions
- Programming Language: Python
- *License*: GPL V3
- *URI*: http://urbanmodel.asu.edu/popgen.html
- *Organisation*: Arizona State University

4.2.1.3 SynthPop

- *Description*: Reimplementation of PopGen using the modern scientific Python stack, with a focus on performance and code reusability
- *Functionality*: Similar to PopGen
- Programming Language: Python
- License: BSD
- *URI*: https://github.com/UDST/synthpop
- *Organisation*: Paul Waddell, Synthicity, now Autodesk

4.2.2 Software for multi-agent simulations

4.2.2.1 MATSim

- *Description*: MATSim provides a framework to implement large-scale agent-based transport simulations (Grether and Nagel (2013)).
- *Functionality*: Agent-based simulation for public transport. Requires transportation network data, human population data and transportation survey data. Generating synthetic human population seems not to be handled by MATSim. MATSim creates agent activity plans from activity chains, which represent round-trips (Rieser et al. (2006)).
- Programming Language: Java
- *License*: GPL v2
- *URI*: http://www.matsim.org
- *Organisation*: TU Berlin, ETH Zurich, Senozon

4.2.2.2 MITSIM

- *Description*: MITSIM is the traffic simulator that is part of MITSIMLab. MITSIMLab is an open-source simulation-based laboratory that was developed for evaluating the impacts of alternative traffic management system designs at the operational level and assisting in subsequent design refinement. MITSIMLab was used by KTH to model traffic in Stockholm.
- *Functionality*: MITSIM requires a model of the road network and origin-destination-time triples. Driver behaviour parameters are assigned to each agent. MITSIM focuses on road traffic and supports dynamic traffic assignment (DTA), where agents can adapt their behaviour dynamically based on the observed conditions on the road (Antoniou et al. (2011)).
- Programming Language: C++
- License: MIT
- *URI*: https://its.mit.edu/software/mitsimlab
- *Organisation*: Intelligent Transportation Systems Lab at MIT

4.2.2.3 UrbanSim

- *Description*: UrbanSim is a model system for analysing urban development (Waddell (2011)). UrbanSim is part of Urban Data Science Toolkit (UDST).
- *Functionality*: Agent design (based on city planning, regulation, transportation network etc.), simulation. Generating synthetic population is handled by another component of UDST called synthpop (see previous section).
- *Additional information*: UrbanSim was initially implemented in Java, then reimplemented in Python, then reimplemented again in Python to make use of tools developed by the PyData community.
- *Programming Language*: Python, using PyData tools
- *License*: 3-clause BSD, it's possible that proprietary parts exist too.
- *URI*: https://github.com/UDST/urbansim
- *Organisation*: Paul Waddell, Synthicity, now Autodesk

4.2.2.4 MUSSA II

- *Description*: MUSSA II is a model designed to forecast the expected location of agents, residents and firms, in the urban area, based on the paradigm of static market equilibrium [29].
- *Functionality*: The system is based on a simulation that reaches a fixpoint. Agent design and this kind of simulation is what the system provides. Agent design and simulation take into account regulation, budgets of agents, accessibility and preferability of sites.
- Programming Language: Unknown

- *License*: Code doesn't seem to be available
- *URI*: http://link.springer.com/chapter/10.1007%2F978-3-642-12788-5_5
- *Organisation*: University of Naples Federico II, University of Southampton

4.2.2.5 Simdemics

- *Description*: general-purpose modelling environment to support pandemic planning and response.
- *Functionality*: scalable to networks with 300 million agents, interactive capabilities, extended with additional components for Internet access, data analytics, etc. HPC ready.
- *Programming Language*: apparently Charm++
- *License*: not available
- *URI*: not available
- *Organisation*: Virginia Tech
- *Note*: Simdemics and its components have been described in several articles (e.g., [30], [31], [32], [33], [34]) but the descriptions are neither complete, nor consistent. For example, in an overview article for *Computing in Science & Engineering* [35], Pam Frost Gorder says "Simdemics has three variants: EpiSims, EpiSimdemics, and EpiFast, which let users trade off between model generality and processing speed". In fact, EpiSims and EpiFast appear to be predecessors of Simdemics, while EpiSimdemics is described as "an algorithm" in [34], but also as "simulation" and an "implementation" (in the same article!), and as an "agent-based simulation method" in [31]. The article [31] starts by saying

  > "In this paper, we describe an interaction-based highly resolved modeling approach for representing and reasoning about large scale epidemics. We build on our earlier work (Barrett et al. 2008), wherein we described EpiSimdemics [...] Specifically, we developed *Simdemics*, that in conjunction with EpiSimdemics, allows us to study the joint evolution of policy, disease dynamics, human behavior and social networks as an epidemic progresses."

  and concludes with

  > "We have presented EpiSimdemics, a large scale, highly detailed, interaction-based epidemic simulation."

  (there is no mention of Simdemics in the concluding section of the paper).

4.2.2.6 FRED

- *Description*: FRED (A Framework for Reconstructing Epidemiological Dynamics) is a system that models the spread of infectious diseases, and effects of response strategies using multi-agent simulation [28].
- *Functionality*: Multi-agent simulation of spreading of infectious diseases, which takes into account geographic locations and social networks. Multiple evolving

strains of pathogens can be simulated, as well as response and mitigation strategies. FRED requires a synthetic population provided by a 3rd party tool.

- Programming Language: C++
- *License*: 3-clause BSD
- *URI*: http://fred.publichealth.pitt.edu
- *Organisation*: Graduate School of Public Health, University of Pittsburgh
- *Note*: FRED is one of the simplest simulation systems based on synthetic populations. Almost all essential aspects, such as agents' features or activity patterns are hard-coded. The source code base is small, currently under 45000 lines of C++ code. In a certain sense, it is the complete opposite of Simdemics. Nevertheless, FRED and Simdemics have been used in very similar studies, for example in the analysis of policies to reduce the impacts of influenza [36] using FRED versus [37] and [30], which use Simdemics).

4.2.2.7 ABM++

- *Description*: software framework for implementing agent based models using C++ [38]
- *Functionality*: allows applications to run on distributed architectures, provides a C++ message passing API, a serialisation interface that allows objects to be moved between distributed nodes, a synchronization method and both time-stepped and distributed discrete event time update mechanisms
- Programming Language: C++
- License: GPL
- *URI*: http://parrot-farm.net/ABM++/
- *Organisation*: Douglas Roberts (RTI International)
- HPC capabilities: yes

4.2.2.8 Pandora

- *Description*: an agent-based modelling framework for large-scale distributed simulations [39]
- *Functionality*: full Geographical Information System support, support for parameter sweeps, GUI tool (*Cassandra*) used to analyse the results generated by a simulation
- *Programming Language*: C++, language bindings to Python
- *License*: GPL v3
- *URI*: http://xrubio.github.io/pandora/
- *Organisation*: Xavier Rubio-Campillo (Barcelona Supercomputing Centre)
- HPC capabilities: yes

4.2.2.9 Repast

- *Description*: Repast Simphony (REcursive Porous Agent Simulation Toolkit) is a system for multi-agent simulation especially focused on applications in social sciences [40].
- *Functionality*: Repast is a Java-based generic environment for programming multi-agent simulations. In addition to Java, Repast allows model development in Python and C#. Additionally, Repast exists in two more variations: Repast Simphony and Repast HPC. Repast Simphony is a Java-based environment for running simulations on small computing clusters. Repast HPC is a C++ based environment for running simulations on supercomputers [41].
- Programming Language: Java
- *License*: BSD (and other free licenses)
- *URI*: http://repast.sourceforge.net
- *Organisation*: University of Chicago
- *HPC capabilities*: yes (Repast HPC)

4.2.2.10 Swarm

- *Description*: Swarm is a versatile multi-agent simulation system [42]. The framework is generic, but has a steep learning curve [43].
- *Functionality*: Swarm provides a general language and libraries for creating multi-agent simulations. The environment for agents taking part in a simulation is also modelled as a set of other agents, which interact with the proper agents. Observations on the simulation are also performed using special 'observer' agents. Swarm also profides a Java interface, which allows programming the simulations in Java.
- Programming Language: Objective-C
- License: GPL
- *URI*: http://www.swarm.org/wiki/Main_Page
- *Organisation*: Swarm Development Group, Santa Fe Institute

4.2.2.11 Mason

- *Description*: Mason (Multi-Agent Simulator Of Neighborhoods) is a multi-agent simulation system focused at simplicity and speed [44]. A related project DMASON (Distributed Mason) is developed independently.
- *Functionality*: Mason provides a light-weight simulation environment. The simulations yield deterministic results across platforms. Furthermore, running simulations can be checkpointed, and migrated to another machine, and possible another platform.
- Programming Language: Java

- *License*: Academic Free License, version 3.0
- *URI*: http://cs.gmu.edu/~eclab/projects/mason/
- *Organisation*: George Mason University

4.2.2.12 FLAME (Flexible Large-scale Agent Modelling Environment)

- *Description*: FLAME is a generic agent-based modelling system [45].
- *Functionality*: Agents are modelled as finite state machines, broadcast communication method, communication is synchronised. A variant of the framework (FLAMEGPU) [46] allows parallel execution of simulations in HPC environments and on GPUs.
- Programming Language: C
- *URI*: http://www.flame.ac.uk/
- *Organisation*: Science and Technology Facilities Council
- HPC capabilities: yes

4.2.2.13 NetLogo

- *Description*: multi-agent programmable modelling environment
- *Functionality*: supports behaviours, agent lists, graphical interfaces
- *Programming Language*: dialect of Logo (implemented in Scala and Java)
- *License*: GPL v2
- *URI*: https://ccl.northwestern.edu/netlogo/
- *Organisation*: Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
- HPC capabilities: no.
- *Remark*: extensively used in education, large collection of models, high-quality documentation available

4.2.3 Software for complete SIS simulations

4.2.3.1 ILUTE

- *Description*: ILUTE is a microsimulation model of urban systems, which simulates transportation and changes in land use.
- *Functionality*: Includes population synthesis, agent design based on projects, simulation.
- Programming Language: C++
- *License*: Unknown (apparently distributed on a CD)
- *URI*: http://www.civ.utoronto.ca/sect/traeng/ilute/ilute_the_model.htm
- *Organisation*: University of Toronto

4.2.3.2 ILUMASS

- *Description*: ILUMASS (Integrated Land Use Modelling and Transportation System Simulation) is a microscopic simulation model of land use, transport and environmental impacts in urban regions.
- *Functionality*: Includes population synthesis, agent design based on daily activities, and simulation.
- Additional information:
- One fundamental reason for the partial success may be that scientists are often not very well prepared for the computer programming aspects (Wilson 2006a, b). However, there are other areas where the project was simply too inefficient in its work processes
- (from http://www.spiekermann-wegener.de/pub/pdf/PWMW_ILUMASS.pdf)
- ILUMASS is partially based on the IPRUD model, which dates back to 1977.
- Programming Language: Unknown
- *License*: Code doesn't seem to be available
- *URI*: http://www.spiekermann-wegener.de/pro/ilumass_e.htm
- *Organisation*: Seven research institutions in Germany

4.2.3.3 TRANSIMS

- *Description*: TRANSIMS is a complete simulation environment for transportation.
- *Functionality*: Includes population synthesis, agent design and simulation. The simulation is iterated in order to reach a Nash equilibrium. Some parts of the simulation can be run on a parallel cluster of machines.
- Programming Language: C++
- *License*: Apache license 2.0, NASA Open Source Agreement
- *URI*: https://www.fhwa.dot.gov/planning/tmip/resources/transims/
- *Organisation*: Federal Highway Administration, US Department of Transportation
- HPC capabilities: yes

4.2.4 Available software for implementing DSLs

In a very simplified manner we can say that, when defining and implementing a new programming language, one starts with a formally described grammar, which defines the syntactical expressions of the language, and an operational semantics of the language, telling us how the syntactical expressions, when executed, will affect the state of the computer. An interpreter or a compiler for the language will take a source code file, containing expressions in the language, and translate it into a sequence of machine-level instructions which, when run, will produce the specified changes to the computer.

Many general-purpose languages, such as assembly language, C, C++, Java, etc. have been designed in order to support the representation of computer-specific data structures and instructions: individual memory locations, chunks of memory locations, jumps, increments, shifting registers, etc. When it comes to designing high-level data structures and functions, for example government measures, stochastic transition functions, etc., their syntax is somewhat cumbersome. For instance, because a lot of details must be specified, related to how the high-level elements are to be represented in the computer.

When designing a DSL, the aim is to have a representation for the high-level elements as smooth as possible. That is why, in many cases, the syntax of the DSL is not going to simply adopt the syntax of an existing language. In such cases, we need to provide a new compiler, written in a different language, for translating the DSL into machine code. The DSL is then called *stand-alone*.

There are, however, programming languages whose syntax does turn out to be suitable for defining the high-level data structures and functions of the DSL. We can then use the compiler of the existing languages directly for translating the DSL into source code. In that case, the DSL is called *embedded*.

The rest of this section is divided along the distinction between stand-alone and embedded DSLs. In the first case, we need software tools for constructing a new interpreter or compiler. In the second case, we need a programming language which is suitable for expressing high-level elements in a convenient fashion.

4.2.5 Software for stand-alone DSLs

4.2.5.1 Flex

- *Description*: open source generator of lexical analysers
- *Functionality*: takes as input a rule file and generates C code that lexes source files according to the rules
- Programming Language: C
- *License*: BSD-like
- *URI*: http://flex.sourceforge.net/
- *Organisation*: actively supported by Will Estes

4.2.5.2 Bison

- *Description*: general-purpose parser generator

- *Functionality*: converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser
- Programming Language: C
- *License*: GNU 3 or higher
- *URI*: https://www.gnu.org/software/bison/
- Organisation: FSF

4.2.5.3 ANTLR

- *Description*: parser generator
- *Functionality*: covers the functionality of both Flex and Bison
- Programming Language: Java
- *License*: three-clause BSD license
- *URI*: http://www.antlr.org/
- *Organisation*: maintained by Terence Parr, University of San Francisco.

4.2.5.4 Parsec

- *Description*: monadic parser combinatory library
- *Functionality*: provides combinators (simple higher-order functions) to create parsers from grammars
- Programming Language: Haskell
- *License*: BSD-style
- *URI*: https://github.com/aslatter/parsec

4.2.6 Software for embedded DSLs

4.2.6.1 Lisp

- *Description*: a family of languages related to the language designed by John McCarthy in the late '50s
- *Functionality*: Lisp was originally designed for AI tasks, including symbolic processing and self-modifying code. Modern implementations offer powerful facilities for manipulating program representations.
- *Availability*: many available high-quality open-source implementations
- *URIs*: Unknown

4.2.6.2 Haskell

- *Description*: lazy pure functional language
- *Functionality*: high-level type system, pattern-matching, appealing syntax
- *Availability*: currently one high-quality open-source implementation, GHC (Glasgow Haskell Compiler)
- *URIs*: https://www.haskell.org/

4.2.6.3 Agda

- *Description*: lazy dependently-typed programming language
- *Functionality*: mature dependently-typed system, appealing high-level syntax (more flexible than Haskell's), suitable for specifying not just the high-level elements, but also their properties
- *Availability*: one open-source implementation, evolving, not suitable for industrial-level programming
- *Remark*: within CoeGSS, Agda could be used in conjunction with other tools in order to ensure correctness of implementations
- *URI*: http://wiki.portal.chalmers.se/agda/pmwiki.php

4.2.6.4 Idris

- *Description*: eager dependently-typed programming language
- *Functionality*: similar to Agda's, type system is somewhat less mature, aims at being usable for systems programming. Special facilities for DSLs.
- *Availability*: one open-source implementation, relatively unstable, not suitable for industrial-level programming
- *Remark*: could play the same role as Agda in CoeGSS
- *URI*: https://github.com/idris-lang

5. Type theoretical approaches to the representation of uncertainty

5.1 Introduction

In every project and implementation that involves the processing of large amounts of data it is essential to consider ways of dealing with incompleteness, inconsistency and lack of accuracy, i.e. with several kinds of uncertainty in data.

However, not only the uncertainty in data has to be considered but also the uncertainty in computation, i.e. the reliability of deployed software components.

In this chapter we describe the starting point of an approach for representing different kinds of uncertainty based on type theory. The outcome of this type theoretical approach will on the one hand contribute to the development of DSLs for synthetic populations and agent based modelling systems. On the other hand, the validated implementation of methods of interval analysis in a functional language will lead to touchstones for the correctness of components of GSS- simulations.

In section 6.2 we give an overview on several kinds and sources of uncertainty. In section 6.3 the role of functional programming languages as host languages for DSL is described and in section 6.4 we identify the kinds of uncertainties we have to consider when synthetic populations will be created.

Section 6.5 describes monads as a key concept for uncertainty. Finally, section 6.6 lists existing monadic implementations of different kinds of uncertainty.

5.2 Uncertainty

Census data, data from measurements, statistical analyses and numerical simulations are often incomplete, inconsistent or inaccurate.

Web, social media, sensor networks and company data are often affected by different kinds of uncertainty. Uncertainty can arise because of inaccurate sensors. Data from social media can be noisy, enterprise databases incomplete, outdated or simply wrong. In models of socio- economic systems, certain notions can turn out to be ambiguous. Typical examples are notions of stability, resilience, sustainability, viability.

Further uncertainties can arise in processing of data, for example when machine learning techniques are applied. This concerns the ability of clustering, regression, and correlation.

While data from governmental sources like Census are well documented, those from other sources like social media or sensors are not. They often lack in information about provenance and are unlikely to have been sampled according to any recognized sampling scheme.

To make rational decisions, we need tools to reason about data uncertainty, understand how uncertainty propagates from data to simulations and communicate our findings to decision makers.

As the software, to be integrated or from scratch developed, has to deal with uncertainties, we need to provide a feasible solution which can mitigate negative impacts when propagating uncertainties from data to simulation results.

5.3 Functional Programming languages and DSLs

DSLs support solving domain specific problems in terms of the concepts of the domain itself. They lead to programs that are easy to understand and reason about. They help to improve the productivity of domain experts. These do not need to be programming experts, as the syntax can be specifically designed for them. They can take advantage of the DSL to implement models in terms of domain specific notions and constructs.

The most popular way of implementing a DSL is to embed it in a general purpose programming language, called the host language. It is efficient, because the resulting DSL only needs to contain the concepts and functions needed for the specific domain. Moreover, embedded DSLs are easily extensible and customizable to particular problem instances.

Functional programming languages have proven to be good host languages, mainly because their type systems support algebraic data types and pattern matching. This makes it easier to reason about programs and prove properties.

Referential transparency, sound type checking and termination checking can be extremely useful when it comes to show that a model implementation meets its specification, that is, it is correct. Thus functional programming languages as host languages for DSLs play a crucial role in developing provably correct models.

5.4 Synthetic Populations and uncertainty

To properly represent uncertainty in a DSL, we have to understand which kinds of uncertainty affect the data and the models we deal with.

For implementing an agent based modelling system on the basis of synthetic populations we consider the steps detailed in Chapter 6 of this document.

The first step consists of setting up of the (static) data characterizing the agents, for example in terms of age, height, gender, income and spatial data. Depending on the source of the real world data that we will use as the blueprint for the synthetic population, we may want to pair the raw data with measures of precision or reliability. For example, we might want introduce features like disease risk or a range of income instead of a single salary number.

In steps 2 and 3 where the transition functions (i.e. activity patterns) are defined and adapted, we plan to deploy learning techniques to find the patterns. This is a highly nondeterministic process and creates much uncertainty of the resulting data. It results not only in probabilities of state transitions but, depending on the quality of the data (errors in measurement, the sampling techniques used for a study we learned from, etc.) such probabilities will be affected with some reliability measure, that has to be taken into account when the results of the simulations are evaluated.

With the help of a provable correct implementation of interval arithmetic we would be able to qualify the results of the software components used for the parameter estimation of the activity patterns and the synthesis of best-fitting synthetic populations.

When running simulations with nondeterministic activity patterns, each step yields a set of possible outcomes, perhaps with different probabilities. Typically, different kinds of uncertainties have to be tracked down and propagated correctly in order to estimate their cumulative effect on the outcome of the simulation.

5.5 Monads

A ubiquitous structure in functional programming languages is that of a monad. Many common programming concepts can be described in terms of a monadic structure, including side effects such as input/output, variable assignment, exception handling, nondeterminism and concurrency. Monads are in particular useful to represent uncertainties. In fact they may be considered as a kind of metastructure that contains everything that we would expect from an uncertainty related notion.

Formally a monad *M* is a type constructor *M: Type → Type* satisfying the following properties:

M is a *functor*, that means there exists a polytypic function *fmap M* that maps a function *f: A → B* producing outputs of type B from inputs of type A, to a function *fmap_M f*: *M A → M B*, that produces outputs of type *M B* for inputs of type *M A.*

There is a way to represent certain outcomes as corner cases of uncertain events. This is given in terms of a function certain_M: A → MA.

There is a way to reduce "second order" uncertainty provided by a function

Reduce_M: M(M A) → M A

Moreover, *fmap_M, certain_M* and *reduce_M* have to fulfil a small number of equations, the *monad laws*, for *M* to be a monad. Thus, defining a monad M means implementing *fmap_M, certain_M* and *reduce_M,* and showing that these functions fulfil the monad laws.

The behaviour of the agents in a synthetic population may be seen as a *sequential decision procedure* [47]. This is a process described by a sequence of states. At every time step the agent is in a certain state (e.g. healthy or not) and after choosing some control (making a decision) and depending on its environment it enters a new state.

In the following sections we describe some structures that have been or will be implemented as monads: lists, fuzzy sets, probability distributions, rough sets and intervals.

5.5.1 Identity monad

A deterministic decision process can be described, at each decision step, by a set of possible states, a set of possible controls and a transition function [47]. For every state x and selected control y, the transition function *step* calculates the next state x' of the process. More formally we describe a deterministic process in terms of state and control functions

X: (t : Nat) → Type and Y: (t: Nat) → (x: X t) → Type and a transition function step: (t : Nat) → (x : X t) → (y : Y t x) → X (t+1). The interpretation is that step t x y

is the outcome of selecting control y in x.

A deterministic process may be viewed as a nondeterministic process. The corresponding monad is the identity monad *Id*, with *id A=A* for every type *A*, and with *map_Id f=f, certain_Id = reduce_Id =id.*

5.5.2 Lists

In a nondeterministic decision process, the selection of a control does not determine a unique next state. Instead, there is a whole set of possible next states. These can be represented with a list.

If we adopt the Haskell *List A* for the type of lists of elements of type *A*, the step function for a nondeterministic decision process has the type:

step': (t: Nat) → (x:X t) → (y:Y t x) → List (X(t+1))

A list type is supported by every functional programming language. The function *certain_List* produces a singleton list from the given element and *reduce_List* produces a list when given a list of lists as input by concatenation.

5.5.3 Probability distributions

Sometimes we know that, when selecting a control y in a given state x at decision step t, certain next states are more probable than others. In this case we take advantage of the order in *step t x y: List (X(t+1))* to represent this knowledge. For instance, we can put states with higher probability first. Sometimes we even know numerical probability values for every possible outcome. If the number of possible outcomes is finite *(X(t+1)* can still be infinite), we can just replace *List(X(t+1))* by finite probability distributions on *X(t+1).* For instance*, Prob(X(t+1))=List(X(t+1), Real).*

If we define *Prob* as a monad the function *reduce_Prob* is supposed to give a combined probability distribution resulting of a sequence of probabilistic choices, while *certain_Prob* interprets a particular value as probability distribution with only one possible event.

5.5.4 Fuzzy sets

The typical mathematical approach to deal with ambiguity is to model vague concepts by means of fuzzy sets. While (crisp) properties usually are modelled with membership functions χ*: X → {0,1}* there are also concepts like "large", "red", "fast" that can best be described with degrees of membership.

A fuzzy set is a function f: X → L, where L represents a lattice. If an object x belongs fully to the concept then f(x) =1_L, if it doesn't have the property at all, then f(x) = 0_L, where 1_L and 0_L are the maximum or the minimum of the lattice respectively.

5.5.5 Rough sets

There is another mathematical concept that has been proven useful for modelling uncertainties. When defining the decision functions for the ABMS we have to derive decision rules for the agents from the given real data. For this knowledge discovery process the concept of rough sets might be suitable. The central notion with rough sets is that of indiscernibility. If we have a set R of objects that cannot be distinguished with the given information we may represent it as a rough set. We describe it by a formal approximation in terms of a pair of sets, its lower and upper approximation. The first describes the set of objects that definitely belong to the set, the second the set of objects that possibly belong to it.

5.5.6 Intervals

Interval arithmetic was developed starting from the 1950ies approach for implementing reliable numerical methods. In interval arithmetic, plain numbers are replaced by intervals. Thus, the result of a computation (as well as its arguments or inputs) is an interval. The interpretation is that each number in that interval is a “possible” result. Implementations of interval arithmetic rely on intervals defined as pairs of machine numbers. Set operations on intervals are combined with interval function evaluations to get algorithms for computing enclosures of sets of solutions.

An interval valued *extension* F of a real valued function f is function from intervals to intervals such that, for all *a,b: Real_machine,*   *f x* is in *F [a,b]* for all *x* in *[a,b].*

In interval arithmetic, every interval [a,b] has a dual nature. It is an alias for *(Real_machine,Real_machine)* and it represents a set [a,b]={x:a≤x≤b}.

It is easy to see that intervals as pairs of machine numbers are monads. In order to develop methods for measuring and quantifying the cumulative effect of uncertainties on computations, a minimal library of interval based functions has to be conceived, implemented and verified. Here we will take advantage of dependently typed languages to express unambiguous specifications (e.g. for function extensions, see above) and derive provably correct implementations. It is the proofs as programs paradigm that ensures the correctness, with an implementation that does not match the specification, the type checker would not accept the program.

There are several books on interval mathematics, for example [48] and [49]. Interval mathematics as well as fuzzy theory were developed to implement approximate reasoning. A book that shows the connection between both is [50]. As a starting point to implement interval arithmetic we need a suitable type theory that may be found for instance in [51].

5.6 Existing Implementations

As explained above, we will use a functional programming language as host language for the DSL. One advantage is that we will be able to derive machine checkable (provably correct) applications. Moreover, applications will be easier to understand, maintain and extend. In functional programming languages functions are functions in the mathematical sense. They are designed to accomplish a specific task and do not rely on an external state. Functions return the same value if they are called twice (or in fact infinitely many times) with the same argument. Thus functions can be composed easily with others and tested separately.

Very important for the choice of the host language is its type system. One way to produce fully validated programs is to develop the program along with its proof of correctness. For this we would need a type system that is capable to express the properties to be proved in the programming language itself. A dependent type system like the one of Agda or Idris would be suitable.

Another important characteristic of the host language is its efficiency. In practice many algorithms can be implemented in lazy functional languages like Haskell as efficiently as in imperative languages. A thorough description of techniques to design efficient data structures may be found in [52].

As shown above, we have to implement monadic systems. We plan to start with the approaches outlined in [53], [54] and [55], an associated Haskell library that “allows exact computation with discrete random variables in terms of their distributions by using a monad” for simple probability distributions.

Built upon [53], Harvey has developed a “probability” library [56] for Idris. This library however does not contain the proofs for the monad laws (it uses floating point numbers to represent the probabilities and for floating point numbers we do not even have associativity of addition). But it could be a valuable starting point for an incremental approach [57] towards correctness.

Besides informing the transition functions of SDPs, monads could also represent the likelihoods of the transition matrices of Hidden Markov Models as demonstrated in [58].

A data type of fuzzy sets and a comprehensive list of functions to manipulate them have been described and implemented in Haskell for example in [59]. Furthermore there is a Haskell library “huzzy” [60].

But as in the case of simple probabilities there is no implementation with a machine checkable proof of the monad laws. One approach for deriving provably correct implementations of fuzzy sets, would be to use rational numbers to represent the membership degrees. A library for rationals (with proofs of basic arithmetic properties) was implemented only recently in Idris [61].

For rough sets, as far as we know, there exists no implementation in any functional programming language. On the theoretical level there is a categorical description of rough sets [62] that might be a starting point for a development of a type theoretical representation.

Although there are a lot of libraries for interval based numeric for many imperative programming languages, there is so far no implementation of an interval type or a library with interval based numeric in any functional programming language.

If we had a provably correct implementation of interval based methods, we could use them to evaluate the correctness of the components of GSS simulations (which could be written in C++ or Java for instance).

Some of the components will include the solving of optimization problems. It is known to be hard to check how good a given implementation of these optimization methods is. If we had a provable correct implementation, then we would know which solutions the C++ programs should find.

Thus a provably correct implementation of interval arithmetic is a way to the develop touchstones for the correctness of GSS simulation software by improving its testability.

6. Hardware and software co-design

The following chapter will first of all give a short introduction about the co-design approach in contrast to the classical design approach. For that we will especially refer to Sanders and Stappers[60] where the difference between these two approaches from a general point of view is analysed in deeper detail.

Once the overview is given and the terminology used throughout this chapter is clarified, an overview over the currently available and with that the, at the moment, proposed methods and mechanisms in co-design is given. From this summary a first layout of the "CoeGSS Co-Design Toolbox" is derived.

6.1 The co-design approach

In [63] the following three features of the classical design approach are given:

- Centred on a product or a technology.
- The designer conceives of and gives shape to products or a technology.
- Focused on designing products. This means, the features of a product.

To identify whether a product is currently designed by the traditional approach, one has first to identify the roles of the different parties participating in the design process and their interaction with each other.

6.1.1 Identification of roles in the design process of a HPC-system

Regarding the design approach of today's high performance computing systems (HPC-systems) these three features can directly be found and the role of the designer of an HPC-system can be assigned.

First of all, the design of a HPC-System is clearly centred around its technology, which means mainly its peak performance in terms of floating point operations per second (FLOPS), its interconnect capabilities in terms of gigabyte per second (GB/s), its storage capacity and capability as well as its Power consumption. This can be seen from the fact, that the HPC-systems installed around the world are still ranked in the so called TOP-500 list[61] according to their sustained performance measured with the LINPACK benchmark.

[1] (1) Co-creation and the new landscapes of design, Elisabeth B.-N. Sanders, Pieter Jan Stappers, CoDesign 2008, Vol.4, Iss.1. pp 5-18.
[61] http://www.top500.org/

Looking to the TOP-500 list, it can be derived who are the designers of nowadays HPC-systems giving shape to their technology. At first Intel, as the supplier of the CPUs in 445 systems out of the top 500, has to be mentioned. Intel directly has to be followed by the HPC-system vendors HP, Cray Inc., Sugon, IBM, SGI, Lenovo and Bull delivering 395 out of the TOP-500 HPC-systems[62].

That the design of HPC-systems is focused on designing products is not as clear as the former two points. For the case of scientific applications discussed here it can in some sense be derived from the following example of the Byte per FLOP ratio in the current x86-64 Intel CPUs. With the Haswell architecture [64] this ratio has dropped to 0.142 byte per floating point instruction for a CPU with 2.5 GHz clock rate[63]. This means (while getting two floating point values with 8 byte length each from the main memory) the processor can do 56 cycles or broken down to one byte the FLOP per byte ratio is 7. From this fact most scientific simulation codes suffer since the algorithms simply do not deliver the number of floating point operations which have to be applied to the data in order to satisfy the instructions per byte ratio.

Looking at the CPU development of the last years the drop of the Byte per FLOP ratio was accelerated by two facts:

(1) The development of the memory clock rate could not keep track with the development of the CPU clock rate.
(2) Due to the problem that the increase of the CPU clock rate is no longer easily achievable, the performance of nowadays CPUs is increased by putting more cores on the die.

This development finally ended up in the fact, that since 2014 an additional ranking for HPC systems was established due to the problem, that the Linpack benchmark [65] performance results of the TOP-500 list degenerated to artificial numbers, no longer achievable with real world applications. This ranking is done by the HPCG benchmark[64] which measures the performance of the systems with a conjugate gradient method, by that being much closer to real scientific simulation applications in its code features and also in the performance it achieves on modern HPC-Systems. Looking at the HPCG results[65] one can see that only two systems out of the top 10 are able to achieve more than 2% of their peak performance in this real world benchmark.

[62] Data based on the TOP-500 list retrieved on Nov. 30th 2015
[63] http://ark.intel.com/products/81908/Intel-Xeon-Processor-E5-2680-v3-30M-Cache-2_50-GHz
[64] http://www.hpcg-benchmark.org/index.html
[65] http://www.hpcg-benchmark.org/custom/index.html?lid=155&slid=282

In conclusion one can understand, that seen from the point of view of scientific simulation application developers and users, aiming to facilitate HPC systems, the statement, "The design of HPC-systems is focused on designing products" can be confirmed. This simply reflects the struggle of the application developers and users to keep track, in the development of the performance of their applications, with the increasing CPU performance in spite of the dropping byte per flop ratio.

With these considerations in mind, it becomes clear that the role of the HPC system's users in the design process is combined of two groups as there are, on the one hand, the scientific simulation code developers and, on the other hand, the simulation code users, deploying the application to the HPC systems to solve their problems of a given kind. In many cases the code developers are also the code users but, for the sake of generality, we will keep these two groups separately within the role of the users in the design process.

The third group which, according to Sanders and Steppers, is involved in the design process are the researchers and / or scientists. In the classical design approach this group develops theories on how a potential product should be designed or on how a given product should be used.

In the first case, the theoretical developments are driven by the users who are faced with given products and the researchers try to develop theories on how to improve these products according to the user's current or upcoming needs.

As stated before, in the second case the theoretical developments are driven by the given product, which means the researchers analyse the product and according to their findings, try to come up with theories on how the features of the product should be used to satisfy either the purpose of the products or the needs of the users in the best possible way.

With respect to the groups, which should be involved in the co-design process within CoeGSS, the scientists are currently represented by the staff of the participating HPC-Centres. In Future, once the algorithms and numerical methods to be used for the different GSS-scenarios are sorted out, it seems to be meaningful, to further extend the group of scientists to include experts for numerical mathematics, theoretical algorithm development, compiler theory and HPC-tools development.

With the identified roles in the design process we can now take figure 3 from (1) and extend it to figure 1, so that it provides a summary of what was stated before.

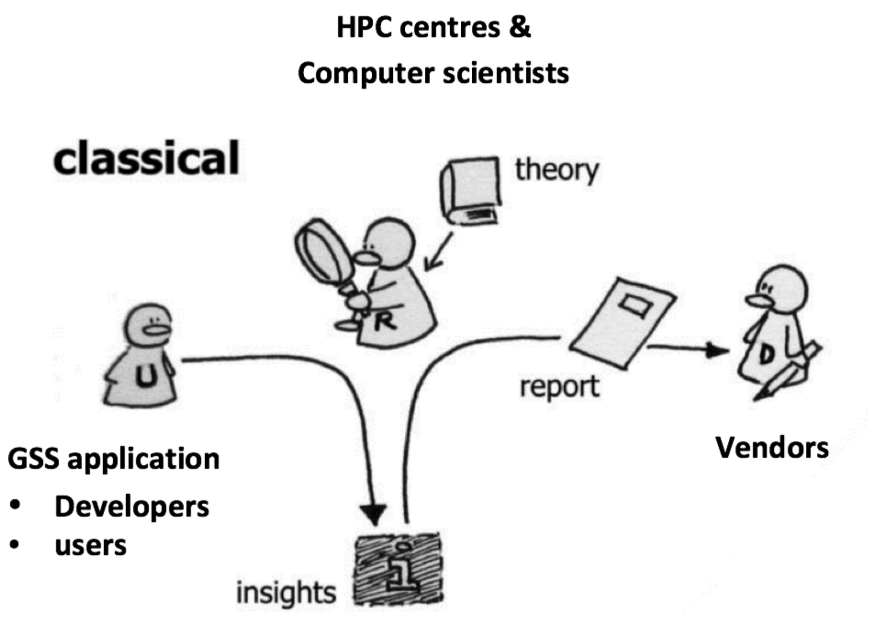

Figure 10. Roles in the classical design approach

6.1.2 Towards co-design of a GSS ready HPC-System

As stated in the previous part for the classical design approach, for the co-design approach [63] defines also four essential features. According to [63] the co-design approach:

- has to be centred around user's needs
- has to be focused on designing for a purpose
- needs longer views and addresses larger scopes of inquiry
- requires the change of roles in the design process

That the design of the HPC-systems facilitated for GSS are designed according to these four features has to be ensured by the CoeGSS co-design toolbox.

To enable the **first feature** of co-design it has to be clear, what are the user's needs or more precise in the given case, what are the user's needs in HPC and HPC features. As this is currently under evaluation, only some very general and basic requirements have already become clear since the project start.

1. HPC-Frameworks for agent based simulations

As a still ongoing literature review and survey among the project partners, especially the use case providers, showed, a central simulation paradigm that is used in GSS are agent based simulations.

2. Analysis tools for large scale heterogeneous datasets

   Since one very basic and also crucial part of an agent based simulation is the modelling of the agent's state transfer function that represents the behaviour of a single agent in a given situation, especially for social interactions and decisions of the agents according to environmental influences and social atmospheres they are placed in, it will be essential to analyse and interpret sociological data from different sources and of different type.

3. Fast external connections and accordingly high incoming bandwidth

   The third requirement arises directly from the second one, as it cannot be assumed, that the large scale heterogeneous data sets to be analysed are located directly inside the analysing HPC-system.

This list is only preliminary. It has to be extended and worked out in more detail during the next months to evolve towards a requirements catalogue that can be taken into account as a starting point once the design of a HPC-system intended to be facilitated by GSS applications is under discussion in a co-design process.

That the focus of the system design has to be centred at GSS applications seems to be quite obvious especially since the centre point of the co-design process is intended to be located at CoeGSS. The concentration of GSS and HPC expertise along with the network that will be established by CoeGSS between the two disciplines and their connected fields will ensure that the **second feature** of a well-defined co-design process, the design for a purpose, is fully regarded.

Enabling the **third feature** of co-design defined by (1) is probably the most difficult task to be accomplished within the three years' duration of the CoeGSS project. But with the vision to turn CoeGSS in a long term basis for GSS on HPC one can definitely think of a growing knowledge database that not only includes user requirements, but also measures how these requirements fit to current systems, how the vendors and computer scientists estimate the impact of coming developments upon these requirements and also measures, that give an idea about how the features of the CoeGSS co-design toolbox influenced the system design and the design process.

The **fourth feature** of a co-design approach as stated by (1) is from the author's point of view the most essential one since it is absolutely necessary to change especially the roles of users and designers in the design process of a GSS ready HPC-system. The first step towards this goal will be to create the awareness for the problems and requirements of the respective other side and of course the awareness for the CoeGSS co-design toolbox among GSS users and the vendors. Only if this awareness can be created, the outcome of the process transformation of the, with the statements made above, identified classical design process of nowadays HPC systems, towards a co-design process of GSS ready HPC-systems, can be achieved.

As a visual summary of the targeted outcome of this design process transformation again an extended version of figure 3 from [63] can be seen in Figure 11.

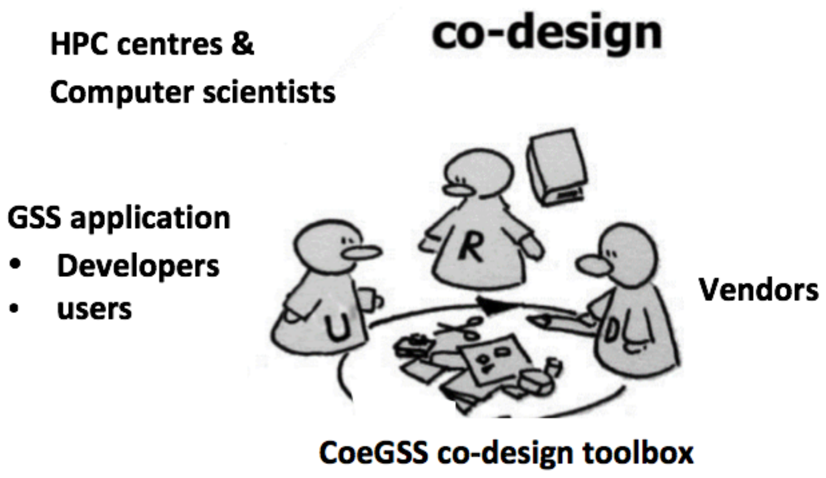

Figure 11. Co-design approach

6.2 Currently available and proposed methods and mechanisms in co-design

As stated in the previous part, the first and most essential mechanism in co-design is the awareness creation among the groups participating in the design process for the problems, needs, tools and methods of the respective other groups participating in the design process.

6.2.1 GSS-awareness creation on the vendors and computer scientists' side

To spread the word about GSS as a field that is now approaching the world of HPC applications and generate awareness for the topic among the HPC community, workshops and conferences are the most suitable mechanism. The CoeGSS Portal will support this awareness creation by guiding stakeholders about any event that could be of interest for the GSS and the HPC communities while, at the same time, it will facilitate access the publication of guides, white papers and training material related to these topics.

A list of the events potentially to target is stated below.

- Workshop for sustained simulation performance

  The Workshop on Sustained Simulation Performance is a cooperation project between HLRS and NEC. The workshop is a meeting platform for scientists, application developers, international experts and hardware designers from different continents to discuss the current state and future directions of supercomputing. This includes future hardware architectures, the future style of programming and directions to highest sustained application performance.

  Scientists working in various application areas will present their recent results and they will share their views to current supercomputer architectures.

  See also: http://www.teraflop-workbench.de

- HLRS/hww Workshop on Scalable Global Parallel File Systems

  The workshop is organised by The High Performance Computing Center Stuttgart, HLRS, and the Höchstleistungsrechner für Wissenschaft und Wirtschaft mbH, hww.

  The workshop presented the major issues and developments associated with the evolution of non-volatile memory and other storage technologies and their influences on future high performance storage systems leading to "The non-volatile challenge". This covered all hardware and software solutions as well as envisaged approaches to solve the issues of handling future amounts of data in the most efficient and economical way and included the challenges of big data analytics, analysis and discovery as well as the whole data life cycle from data generation to data archival especially on high performance systems.

- International Super Computing Conference tells the following:

  In 1986 Professor Dr. Hans Werner Meuer, director of the computer centre and professor for computer science at the University of Mannheim (Germany) co-founded and organized the "Mannheim Supercomputer Seminar" which had 81 participants [63]. This was held yearly and became the annual International Supercomputing Conference [66] and Exhibition (ISC). The conference is attended by speakers, exhibitors, and researchers from all over the world. Since 1993 the conference has been the venue for one of the twice yearly TOP500 announcements where the fastest 500 supercomputers in the world are named.

- ACM/IEEE Supercomputing Conference

  The ACM/IEEE Supercomputing Conference was established in 1988. The annual SC conference continues to grow steadily in size and impact each year. Approximately 5,000 people participate in the technical program, with about 11,000 people overall.

  SC has built a diverse community of participants including researchers, scientists, application developers, computing centre staff and management, computing industry staff, agency program managers, journalists, and congressional staffers. This diversity is one of the conference's main strengths, making it a yearly "must attend" forum for stakeholders throughout the technical computing community.

  The technical program is the heart of SC. It has addressed virtually every area of scientific and engineering research, as well as technological development, innovation, and education. Its presentations, tutorials, panels, and discussion forums have included breakthroughs in many areas and inspired new and innovative areas of computing.

6.2.2 HPC-awareness creation on the GSS-community and GSS-application developers and users side

To create the awareness for HPC, its possibilities, its methods and also its limits among the GSS-community and GSS-application developers’ workshops, conferences but also training are the first methods to target since it is an essential step from using and programming an application for a multicore workstation towards the usage and application programming for HPC-systems.

As in the previous case, the CoeGSS Portal will aim at articulating the awareness creation through features which facilitate the publication and access of GSS and HPC content, such as white papers, success stories, training, etc…

To connect the GSS-community with the HPC-systems’ world the events already mentioned above for the awareness creation on the HPC-community side are also appropriate, which are:

- Workshop for sustained simulation performance
- HLRS/hww Workshop on Scalable Global Parallel File Systems

- International Super Computing Conference [67]
- ACM/IEEE Supercomputing Conference

To get more detailed ideas communicated into the GSS-community the following courses and workshops can be targeted. First of all for parallel programming methods in general, second of all introductions and usage techniques for special systems and last but not least courses that teach their participants how to measure and raise the performance of their codes. Below a few courses of these three classes are mentioned.

- Parallel Programming (MPI, OpenMP) and Tools

  The focus is on programming models MPI and OpenMP. Hands-on sessions (in C and FORTRAN) will allow users to immediately test and understand the basic constructs of the Message Passing Interface (MPI) and the shared memory directives of OpenMP. The last part is dedicated to tools. This course provides scientific training in Computational Science, and in addition, the scientific exchange of the participants among themselves. It is organized by ZIH in collaboration with HLRS. (Content Level: 70% for beginners, 30% advanced)

- Parallel Programming Workshop (MPI, OpenMP and advanced topics)

  The course is organized in three sections which have the following content:

  Distributed memory parallelization with the Message Passing Interface MPI (Mon+Tue, for beginners):

  On clusters and distributed memory architectures, parallel programming with the Message Passing Interface (MPI) is the dominating programming model. The course gives a full introduction into MPI-1. Further aspects are domain decomposition, load balancing, and debugging. An MPI-3.1 overview and the MPI one-sided communication is also taught. Hands-on sessions (in C and FORTRAN) will allow users to immediately test and understand the basic constructs of the Message Passing Interface (MPI).

  Shared memory parallelization with OpenMP-4.0 (Wed, for beginners):

  The focus is on shared memory parallelization with OpenMP, the key concept on hyper-threading, dual-core, multi-core, shared memory, and ccNUMA platforms. This course teaches shared memory OpenMP parallelization. Hands-on sessions (in C and Fortran) will allow users to immediately test and understand the directives and other interfaces of OpenMP. Race-condition debugging tools are also presented.

  Advanced topics in parallel programming with MPI-3.1 (Thu+Fri):

Topics are MPI-2 parallel file I/O, hybrid mixed model MPI+OpenMP parallelization, MPI-3.0 and 3.1, parallelization of explicit and implicit solvers and of particle based applications, parallel numerics and libraries, and parallelization with PETSc. MPI-3.0 introduced a new shared memory programming interface, which can be combined with MPI message passing and remote memory access on the cluster interconnect. It can be used for direct neighbour accesses similar to OpenMP or for direct halo copies, and enables new hybrid programming models. These models are compared in the hybrid mixed model MPI+OpenMP parallelization session with various hybrid MPI+OpenMP approaches and pure MPI.

Hands-on sessions are included on all days. This course provides scientific training in Computational Science, and in addition, the scientific exchange of the participants among themselves.

- Introduction to Hybrid Programming in HPC

Most HPC systems are clusters of shared memory nodes. Such SMP nodes can be small multi-core CPUs up to large many-core CPUs. Parallel programming may combine the distributed memory parallelization on the node interconnect (e.g., with MPI) with the shared memory parallelization inside of each node (e.g., with OpenMP or MPI-3.0 shared memory). This course analysesanalyses the strengths and weaknesses of several parallel programming models on clusters of SMP nodes. Multi-socket-multi-core systems in highly parallel environments are given special consideration. MPI-3.0 has introduced a new shared memory programming interface, which can be combined with inter-node MPI communication. It can be used for direct neighbor accesses similar to OpenMP or for direct halo copies, and enables new hybrid programming models. These models are compared with various hybrid MPI+OpenMP approaches and pure MPI. Numerous case studies and micro-benchmarks demonstrate the performance-related aspects of hybrid programming.

Tools for hybrid programming such as thread/process placement support and performance analysis are presented in a "how-to" section. Hands-on exercises give attendees the opportunity to try the new MPI shared memory interface and explore some pitfalls of hybrid MPI+OpenMP programming. This course provides scientific training in Computational Science, and in addition, the scientific exchange of the participants among themselves.

The course is a PRACE Advanced Training Center event.

- NEC SX-ACE - Vectorization and Optimization

In spring 2015, HLRS installed a next generation vector computer, a NEC SX-ACE. The participants learn about the configuration of the NEC SX-ACE system at HLRS and how to use this cluster of vectorising shared memory nodes. One focus is an introduction in vectorization. More experienced users can learn how to optimize programs based on performance measurements. Additional topics are I/O and the optimization of application programs. The first day presents an introduction to the NEC SX architecture and vectorization. The second day morning is focused on usage aspects and the differences between the predecessor system (NEC SX-9) and the new NEC SX-ACE. The afternoon is dedicated to hands-on sessions.

- Cray XC40 Optimization, and Parallel I/O Courses

In August 2015, the Cray XC40 supercomputer Hornet at HLRS was upgraded to a new system named “Hazelhen" featuring 7724 compute nodes, each equipped with two 12 core Intel Haswell processors running at 2.5 GHz. Each node is equipped with 128 GB DDR4 memory and connected to the other nodes through the Cray Aries network. The peak performance amounts to 7.4 PFlops.

In order to help users running efficiently on this new large system, HLRS and Cray offers a workshop consisting of 3 parts.

Cray XC40/HAZELHEN at HLRS:

The first day gives an overview on this new XC40 system. Specialists from Cray will talk about the hardware, best practices, and the new software enhancements, which will allow you to improve your performance on this new machine.

Efficient Parallel I/O:

The second day is dedicated to use parallel IO at scale, which becomes more and more important. High performance computing produces drastically increasing amounts of data, at decreasing time scales. This workshop will cover parallel file systems, efficient data handling, and efficient parallel IO methods.

Optimization on Cray XC40:

The third and fourth day is mainly intended as a "Bring Your Own Code" (BYOC) workshop, and will cover the Cray Programming Environment, scientific libraries and profiling tools. Specialists from Cray will assist profiling and optimizing your programs.

- OpenMP GPU Directives for Parallel Accelerated Supercomputers

  This workshop will cover the programming environment of Cray hybrid supercomputer, which combines multicore CPUs with GPU accelerators. Attendees will learn about the directive-based OpenMP programming model whose multivendor support allows users to portably develop applications for parallel accelerated supercomputers.

  The workshop will also demonstrate how to use the Cray Programming Environment tools to identify CPU application bottlenecks, facilitate the OpenMP porting, provide accelerated performance feedback and to tune the ported applications. The Cray scientific libraries for accelerators will be presented, and interoperability of OpenMP directives with these and with CUDA will be demonstrated. Through application case studies and tutorials, users will gain direct experience of using OpenMP directives in realistic applications. Users may also bring their own codes to discuss with Cray specialists or begin porting.

  This course provides scientific training in Computational Science, and in addition, the scientific exchange of the participants among themselves.

Since this list is not exhaustive, in general most of the courses and workshops listed at the following locations are appropriate to get a more detailed view to HPC, its techniques and methods and its limits:

- Parallel Programming Workshops and Programming Language Courses 2016[66]
- HLRS Training, Courses and Tutorials - Overview and Course List[67]
- PRACE Training Events[68]
- HPC training courses in europe & prace / providers & course lists[69]

6.2.3 Performance and feature analysis of GSS simulation frameworks and tools

The third pillar of an efficient co-design approach beside the awareness creation among the participating groups will be the detailed knowledge about the applications

[66] http://www.hlrs.de/organization/sos/par/services/training/2016/all
[67] http://www.hlrs.de/organization/sos/par/services/training/course-list
[68] https://events.prace-ri.eu
[69] http://www.hlrs.de/organization/sos/par/services/training/hpc-eu

the users want to execute on the HPC-system to be designed. This means not only the necessary features to fulfil the user's needs when system modelling is concerned like e.g. a special kind of agent or the possibility to model a certain behaviour of the agents, but the really detailed ideas about the technical side of the codes bases of the used applications. All parties participating in the co-design process have to be aware what kinds of code features are absolutely essential, how they are potentially implemented and what are the implications of an implementation on the hardware under discussion.

Since most of nowadays software packages are designed for general purpose usage first of all a kind of benchmark suite consisting of the codes considered by the use case providers (WP4) has to be assembled which then will be analysed for features that are really used and executed by the given use cases. Here it will be absolutely essential to have real world examples delivered by the use case providers since experience with benchmarks showed, that normally these cases are stripped down to a limit where only the compute intensive kernels remain. So on the one hand the examples to go into the benchmark suite have to be real world and have to use all features of the codes under evaluation that are really needed by the use case providers, but by code analysis is has to be ensured, that only the used code parts which show essential resource consumption find their way into the CoeGSS co-design toolbox.

This feature analysis has already been started among the use case providers that is, the codes targeted by the use case providers will be analysed for the features really used by the given case. The first code base that will be analysed is the GLEAM Simulator[70]

The analysis will be carried out by the current state of the art tools in code analysis and performance monitoring:

- CrayPat[71] – performance analysis tool designed for Cray XC platforms.

  License: commercial

- VAMPIR[72] – graphical tool for performance analysis.

  License: commercial

[70] http://www.gleamviz.org/simulator
[71] http://docs.cray.com/books/S-2315-52/html-S-2315-52/z1076617932brbethke.html
[72] https://www.vampir.eu/

- VTUNE[73] – tool for CPU & GPU performance analysis, bandwidth, cache and scalability examination is supported.

  License: commercial, trial available

- TotalView[74] - allows to debug and monitor high-scale parallel applications and provides tools for controlling process and threads execution.

  License: commercial

- Valgrind[75] - framework debugging and profiling applications, allows to detect memory management bugs.

  License: GNU license

- DDT[76] - designed for debugging C, C++ and F90 applications, provides support for multi-process and multi-threaded applications.

  License: commercial

As the tools listed above all distributed under proprietary license, in a second step an evaluation will be carried out whether open source tools are capable to return the performance numbers and analyses, the tools listed above are able to deliver.

[73] https://software.intel.com/en-us/intel-vtune-amplifier-xe
[74] http://www.roguewave.com/products-services/totalview
[75] http://valgrind.org
[76] http://www.allinea.com/products/ddt

6.3 CoeGSS Co-Design Toolbox

Based on the statements made above the first layout of the CoeGSS co-design toolbox can be defined as shown by Figure 12.

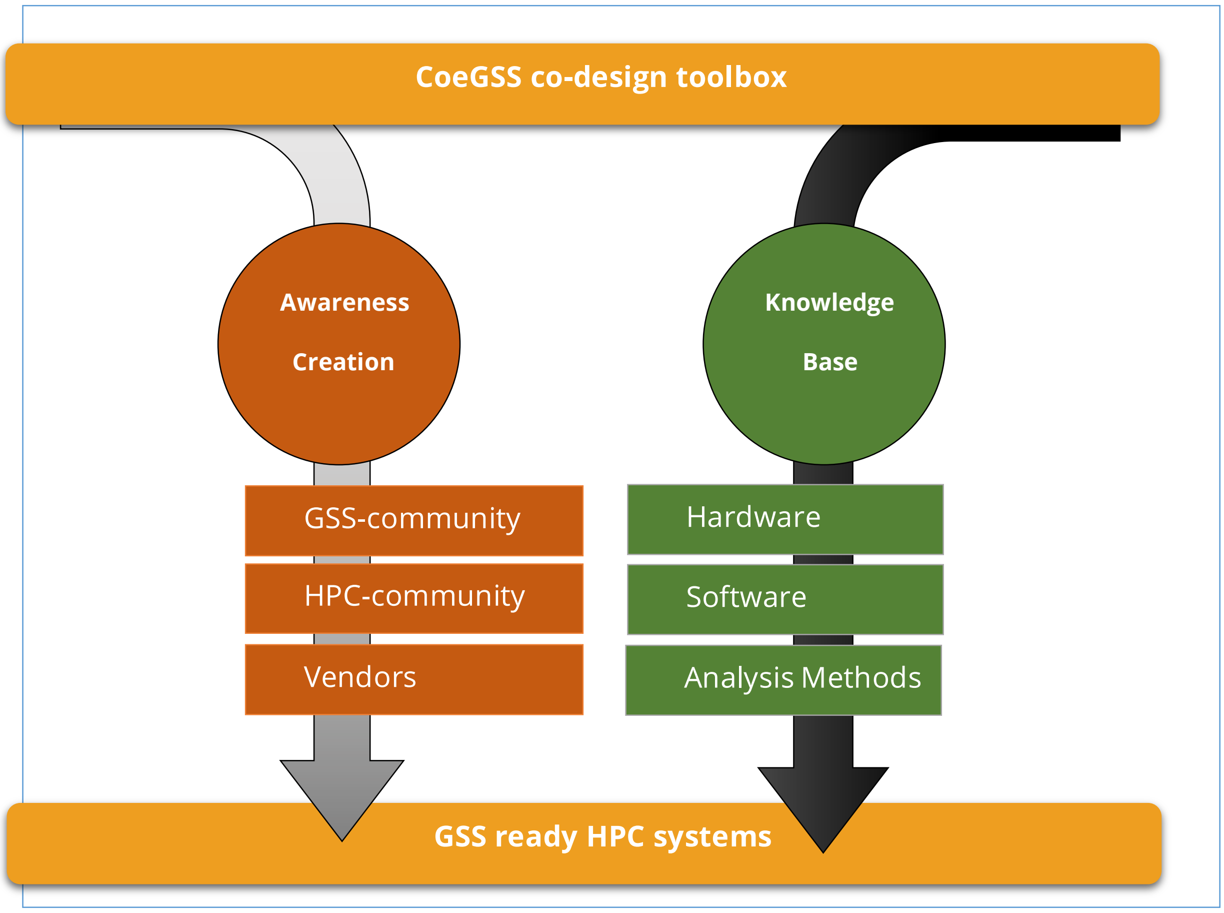

Figure 12. CoeGSS co-design toolbox

7. Summary

This deliverable represents the first official deliverable of work package 3 of the CoeGSS project. It details the work performed during the first four months of the project lifetime. Particular attention has been put on the available methods and tools, which can be (re)used by the CoeGSS project. The main goal of this deliverable is to present state-of-the-art in available mechanisms and software in order to clearly define a starting point in our analysis which allow us to identify gaps and next tasks need in following steps.

In the document we presented tools which can potentially be used for efficient data management and dealing with fault tolerance problems in GSS systems. Remote and immersive visualisation systems which were introduced next are important to explore, investigate and analyse GSS related data. Brief evaluation of available software for creating and running GSS synthetic populations, and for designing and implementing domain-specific languages (DSLs) were presented in the section 5. Next, we presented a type theoretical approach to represent different kinds of uncertainty including a short overview of existing implementations that will become a starting point for the DSLs and for the development of validated numerical methods. These will then help to ensure the correctness of the components of the GSS simulations and to quantify the uncertainty of the output data. Finally, analysis concerning the co-design approach in contrast to the classical design approach was given which is necessary to comply with all requirements stated by GSS.

As it was mentioned before all this works was performed to indicate the starting point in our GSS implementation. In the next steps we will focus on identifying gaps between this and what is needed to provide fully functional GSS system with particular emphasis on specified use cases.

References

[1] B. Tierney, E. Kissely, M. Swany and E. Pouyoul, “Efficient Data Transfer Protocols for Big Data,” Lawrence Berkeley National Laboratory, Berkeley, CA 94270, School of Informatics and Computing, Indiana University, Bloomington, IN 47405, [Online]. Available: https://www.es.net/assets/pubs_presos/eScience-networks.pdf.

[2] ScienceDaily, “Big Data, for better or worse: 90% of world's data generated over last two years,” [Online]. Available: http://www.sciencedaily.com/releases/2013/05/130522085217.htm.

[3] “RDMA over Converged Ethernet,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/RDMA_over_Converged_Ethernet.

[4] “Apache Hadoop,” [Online]. Available: https://hadoop.apache.org/.

[5] “Lustre,” [Online]. Available: http://lustre.org/.

[6] S. M. Hosseini and M. G. Arani, “Fault-Tolerance Techniques in Cloud Storage: A Survey,” *International Journal of Database Theory and Application* , Vols. Vol.8, No.4, pp. pp.183-190, 2015.

[7] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan and D. I. August, “SWIFT: Software Implemented Fault Tolerance,” Departments of Electrical Engineering and Computer Science Princeton University, [Online]. Available: http://liberty.cs.princeton.edu/Publications/cgo3_swift.pdf.

[8] “Berkeley Lab Checkpoint/Restart (BLCR),” [Online]. Available: http://crd.lbl.gov/departments/computer-science/CLaSS/research/BLCR/.

[9] L. Bautista-Gomez, S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama and S. Matsuoka, “FTI: high performance fault tolerance interface for hybrid systems,” in *In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.

[10] J. Ansel, K. Arya and G. Cooperman, “DMTCP: Transparent checkpointing for cluster computations and the desktop,” in *In Parallel & Distributed Processing. IPDPS 2009.*, 2009.

[11] C. Niederauer, “Non-Invasive Interactive Visualization of Dynamic Architectural Environments,” in *SIGGRAPH 2003*, San Diego, Califonia, USA, 2003.

[12] G. Humphreys, “Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters,” in *SIGGRAPH 2002*, San Antonio, Texas, USA, 2002.

[13] D. Rantzau, K. Frank, U. Lang, D. Rainer and U. Wössner, “COVISE in the CUBE: An Environment for Analyzing Large and Complex Simulation Data,” in *2nd Workshop on Immersive Projection Technology (IPT '98)*, Ames, Iowa, 1998.

[14] F. Niebling, R. T. Griesser and U. Woessner, “Using Augmented Reality and Interactive Simulations to Realize Hybrid Prototypes,” in *4th International Symposium on Advances in Visual Computing (ISVC)*, Las Vegas, USA, December 1-3, 2008.

[15] B. e. al., “High Performance Visualisation,” *CRC Press,* 2013.

[16] K. Misegades, “EnSight's Parallel Processing Changes the Performance Equation,” 2002. [Online]. Available: https://www.ceisoftware.com/wp-content/uploads/2012/05/parallel.pdf.

[17] K. Misegades, “Virtual Reality is the Real Deal for Visualization Applications,” 2002. [Online]. Available: https://www.ceisoftware.com/wp-content/uploads/2012/05/vrfeat.pdf.

[18] “Paraview Flavors,” [Online]. Available: http://www.paraview.org/flavors/.

[19] J. Ahrens, B. Geveci and C. Law, “ParaView: An End-User Tool for Large Data Visualization.,” Los Alamos National Laboratory, [Online]. Available: https://datascience.lanl.gov/data/papers/ParaView.pdf.

[20] M. Rivi, “In-situ Visualization: State-of-the-art and Some Use Cases. White Paper.,” 2011. [Online]. Available: http://www.prace-ri.eu/IMG/pdf/In-situ_Visualization_State-of-the-art_and_Some_Use_Cases.pdf.

[21] C. Johnson, S. Parker and D. Weinstein, “Large-Scale Computational Science Applications Using the SCIRun Problem Solving Environment,” in *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, 2000.

[22] S. G. Parker and C. R. Johnson, “SCIRun: A Scientic Programming Environment for Computational Steering,” in *Proceedings of ACM/IEEE Supercomputing Conference (SC)*, 1995.

[23] “SCIRun/BioPSE Documentation,” [Online]. Available: http://scirundocwiki.sci.utah.edu/SCIRunDocs/index.php5/CIBC:Documentation:SCIRun:UserGuide:Concepts.

[24] M. Schirski, “ViSTA FlowLib - a framework for interactive visualization and exploration,” in *Seventh Immersive Projection Technology Workshop, Ninth Eurographics Workshop on Virtual Environments*, Zurich, Switzerland, May 22 - 23, 2003.

[25] T. Kuhlen, T. Beer and A. Gerndt, “The ViSTA Virtual Reality Toolkit,” in *5th High-End Visualization Workshop*, Baton Rouge, Lousianna, USA, 2009.

[26] V. Pascucci, G. Scorzelli, B. Summa, P.-T. Bremer, A. Gyulassy, C. Christensen, S. Philip and S. Kumar, “The ViSUS Visualization Framework,” *High Performance Visualization: Enabling Extreme-Scale Scientific Insight,* 2012.

[27] M. Fowler, “Domain-Specific Languages,” in *Pearson Education*, 2010.

[28] J. J. Grefenstette, S. T. Brown, R. Rosenfeld, J. DePasse, N. T. Stone, P. C. Cooley and W. D. Wheaton, “FRED (a Framework for Reconstructing Epidemic Dynamics): an Open-Source Software System for Modeling Infectious Diseases and Control Strategies Using Census-Based Populations,” in *BMC Public Health 13 (1): 940*, 2013.

[29] F. Martínez and P. Donoso, “The MUSSA II Land Use Auction Equilibrium Model,” in *In Residential Location Choice, 99–113. Springer*, 2010.

[30] C. Barrett, K. Bisset, J. Leidig, A. Marathe and M. Marathe, “An Integrated Modeling Environment to Study the Coevolution of Networks, Individual Behavior, and Epidemics,” *AI Magazine 31 (1),* p. 75, 2010.

[31] K. R. Bisset, X. Feng, M. Marathe and S. Yardi, “Modeling Interaction Between Individuals, Social Networks and Public Policy to Support Public Health Epidemiology,” in *In Simulation Conference (WSC), Proceedings of the 2009 Winter, 2020–2031. IEEE*, 2009.

[32] J.-S. Yeom, A. Bhatele, K. Bisset, E. Bohm, A. Gupta, L. V. Kale, M. Marathe, D. S. Nikolopoulos, M. Schulz and L. Wesolowski, “Overcoming the Scalability Challenges of Epidemic Simulations on Blue Waters,” in *In Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 755–764. IPDPS ’14. Washington, DC, USA: IEEE Computer Society. doi:10.1109/IPDPS.2014.83*, 2014.

[33] K. R. Bisset, J. Chen, X. Feng, Y. Ma and M. V. Marathe, “Indemics: an Interactive Data Intensive Framework for High Performance Epidemic Simulation,” in *In Proceedings of the 24th ACM International Conference on Supercomputing, 233–242. ICS ’10. New York, NY, USA: ACM. doi:10.1145/1810085.1810118*, 2010.

[34] C. L. Barrett, K. R. Bisset, S. G. Eubank, X. Feng and M. V. Marathe, “EpiSimdemics: an Efficient Algorithm for Simulating the Spread of Infectious Disease over Large Realistic Social Networks,” in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, Piscataway, NJ, USA, 2008.

[35] P. F. Gorder, “Computational Epidemiology,” in *Computing in Science & Engineering 12 (1): 4–6*, 2010.

[36] S. Kumar, J. J. Grefenstette, D. Galloway, S. M. Albert and D. S. Burke, “Policies to Reduce Influenza in the Workplace: Impact Assessments Using an Agent-Based Model,” in *American Journal of Public Health 103 (8): 1406–1411*, 2013.

[37] C. L. Barrett, K. R. Bisset, J. Leidig, A. Marathe and M. V. Marathe, “Estimating the Impact of Public and Private Strategies for Controlling an Epidemic: a Multi-Agent Approach,” *In IAAI,* 2009.

[38] D. Roberts, “ABM++ Download Site,” 2013.

[39] X. Rubio-Campillo, “Pandora: a Versatile Agent-Based Modelling Platform for Social Simulation,” in *Proceedings of SIMUL 2014, The Sixth International Conference on Advances in System Simulation: 29–34.*, 2014.

[40] M. J. North, N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen and P. Sydelko, "Complex Adaptive Systems Modeling with Repast Simphony," in *Complex Adaptive Systems Modeling 1 (1): 1–26*, 2013.

[41] C. Nicholson and M. North, "Parallel Agent-Based Simulation with Repast for High Performance Computing," in *Simulation: 0037549712462620*, 2012.

[42] N. Minar, R. Burkhart, C. Langton and M. Askenazi, "The Swarm Simulation System: a Toolkit for Building Multi-Agent Simulations," in *Working Paper 07/1996; 96-06-042. Santa Fe Institute Santa Fe*, 1996.

[43] A. Rob, "Survey of Agent Based Modelling and Simulation Tools," 2011. [Online]. Available: http://www.grids.ac.uk/Complex/ABMS/.

[44] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan and G. Balan, "Mason: a Multiagent Simulation Environment," in *Simulation 81 (7): 517–527*, 2005.

[45] M. Holcombe, S. Coakley and R. Smallwood, "A General Framework for Agent-Based Modelling of Complex Systems," in *In Proceedings of the 2006 European Conference on Complex Systems. European Complex Systems Society Paris, France*, 2006.

[46] M. Kiran, P. Richmond, M. Holcombe, L. S. Chin, D. Worth and C. Greenough, "FLAME: Simulating Large Populations of Agents on Parallel Hardware Architectures," in *In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1-Volume 1, 1633–1636. International Foundation for Autonomous Agents; Multiagent Systems*, 2010.

[47] N. Botta, P. Jansson and C. Ionescu, "A computational theory of policy advice and avoidability," 2015.

[48] R. E. Moore, R. B. Kearfott and M. J. Cloud, Introduction to Interval Analysis, 2009.

[49] G. Alefeld and J. Herzberger, Einführung in die Intervallrechnung, Mannheim/Wien/Zürich: B.I. Wissenschaftsverlag, 1974.

[50] R. Moore, W. Lodwick and R. Albrecht, “Special issue: interfaces between fuzzy set theory and interval analysis,” vol 135 issue 1 of Fuzzy sets and systems, 2003.

[51] W. Tucker, Validated Numerics: A Short Introduction to Rigorous Computations, Princeton: Princeton University Press, 2011.

[52] R. Bird, G. Jones and O. de Moore, “More Haste, Less Speed:Lazy Versus Eeager Evaluation,” *Journal of Functional Programming,* vol. 7, no. 5, pp. 541-547, 1997.

[53] M. Erwig and S. Kollmansberger, “The probability package, version 0.2.5,” hackage.haskell.org/package/probability, 2015-09-03.

[54] M. Erwig and S. Kollmansberger, “Functional Pearls Probabilistic Functional Programming in Haskell,” *Journal of Functional Programming,* 2006.

[55] C. Ionescu, Vulnerability, Doctoral thesis, 2008.

[56] C. Harvey, “probability,” GitHub.com/blackbrane/probability, 2015.

[57] C. Ionescu, “Increasingly Correct Scientific Computing,” in *CICM 2012*, Bremen, 2012.

[58] A. Scibior, Z. Ghahramani and A. D. Gordon, “Practical Probabilistic Programming with Monads,” *Haskell 2015 Proceedings of the 8th ACM SIGPLAN Symposium on Haskell,* pp. 165-176, 2015.

[59] G. Meehan and M. Joy, “Animated Fuzzy Logic,” *Journal of Functional Programming,* vol. 1, no. 1, 1993.

[60] J. Nash, “The hyzzy package,” https://hackage/haskell.org/package/huzzy, 2014.

[61] N. R. Botta, “https://github.com/nicolabotta/SeqDecProbs/tree/master/frameworks/14-,” 2015.

[62] P. Eklund, M. Galan and J. Karlsson, “Categorical Innovations for Rough Sets,” in *Rough Set Theory a True Landmark in Data Analytics*, Berlin Heidelberg, Springer, 2006, pp. 45-69.

[63] E. B.-N. Sanders and P. J. Stappers, “Co-creation and the new landscapes of design,” in *CoDesign*, vol. Vol. 4, 2008, pp. 5-18.

[64] “Haswell (microarchitecture),” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Haswell_%28microarchitecture%29.

[65] “LINPACK benchmarks,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/LINPACK_benchmarks.

[66] “International Supercomputing Conference,” [Online]. Available: https://en.wikipedia.org/wiki/International_Supercomputing_Conference.

[67] “The SC Conference Series,” [Online]. Available: http://www.supercomputing.org/about.php.

[68] C. Osaka, Purely functional Data Structures, 1998.

[69] C. Harvey, “probability,” GitHub.com/blackbrane/probability.

[70] A. Constantinos, H. N. Koutsopoulos, B.-A. Moshe and A. S. Chauhan, “Evaluation of Diversion Strategies Using Dynamic Traffic Assignment,” *Transportation Planning and Technology 34 (3),* p. 199–216, 2011.

[71] “Economic and Social Impact of Influenza Mitigation Strategies by Demographic Class,” *Epidemics 3 (1),* p. 19–31, 2011.

[72] D. Grether and K. Nagel, “Extensible Software Design of a Multi-Agent Transport Simulation,” in *Procedia Computer Science 19: 380–388*, 2013.

[73] M. Rieser, K. Nagel, U. Beuck, M. Balmer, J. Rümenapp and G. G. Rümenapp, “Truly Agent-Oriented Coupling of an Activity-Based Demand Generation with a Multi-Agent Traffic Simulation,” 2006.

[74] P. Waddell, "Integrated Land Use and Transportation Planning and Modelling: Addressing Challenges in Research and Practice," in *Transport Reviews 31 (2): 209–229*, 2011.

[75] S. W. Hodson, S. W. Poole, T. M. Ruwart and B. W. Settlemyer, "Moving Large Data Sets Over High-Performance Long Distance Networks," Oak Ridge National Laboratory, [Online]. Available: http://info.ornl.gov/sites/publications/files/Pub28508.pdf.

[76] "How HPC is Hacking Hadoop," [Online]. Available: http://www.hpcwire.com/2014/02/11/hpc-hacking-hadoop/.

[77] "Adapting Hadoop to HPC Environments," [Online]. Available: http://www.hpcwire.com/2014/02/14/adapting-hadoop-hpc-environments/.

[78] D. Krishnan, M. Tatineni and C. Baru, "myhadoop-hadoop-on-demand on traditional hpc resources," San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego, 2011.

[79] "MyHadoop," [Online]. Available: https://github.com/glennklockwood/myhadoop.

[80] "Running Hadoop Clusters on Gordon," [Online]. Available: http://users.sdsc.edu/~glockwood/di/hadoop-hpc.php.

List of tables

List of figures

List of Abbreviations

DoW	Description of Work
EC	European Commission
EGI	European Grid Infrastructure
CoeGSS	Center of Excellence for Global System Science
ESFRI	European Strategy Forum on Research Infrastructures5
HPC	High Performance Computing
HPDA	High Performance Data Analysis
ToR	Terms of Reference
WP	Work Package