DEEP-HybridDataCloud

INITIAL PLAN FOR USE CASES

DELIVERABLE: D2.1

Document identifier:	DEEP-NA2-D2.1-V14.0.odt
Date:	27/04/2018
Activity:	WP2
Lead partner:	HMGU
Status:	FINAL
Dissemination level:	PUBLIC
Permalink:	http://hdl.handle.net/10261/164311

Abstract

This report summarises the work of WP2 on the initial plan on the selection of Use Cases, providing a key input to the design of the DEEP-HybridDataCloud testbed and laying out the DEEP-HybridDataCloud solutions will be used. This document includes a description of the Research Communities involved and of the use-cases proposed, including Data Management and Computational Intensive issues. Based on the inputs provided by the WP2 partners, an initial list of requirements for each application has been collected, that has been used in turn to assemble a list of common requirements that has been provided as input to the discussion with the technical work packages of the DEEP-HybridDataCloud project.

Copyright Notice

Delivery Slip

	Name	Partner/Activity	Date
From	Wolfgang zu Castell	HMGU / WP2	30/04/2018
Reviewed by	Ignacio Blanquer Fernando Aguilar Álvaro López	UPV CSIC CSIC	23/04/2018
Approved by	Steering Committee		27/04/2018

Document Log

Issue	Date	Comment	Author/Partner
V1.0	26/02/2018	First template version	Álvaro López / CSIC
V2.0	26/03/2018	TOC	Wolfgang zu Castell / HMGU Marcus Hardt / KIT Lara Lloret / CSIC
V3.0	15/03/2018	First version	Marcus Hardt / KIT Lara Lloret / CSIC
V4.0	20/03/208	Methodology updated	Marcus Hardt / KIT Valentin Kozlov / KIT Lara Lloret / CSIC
V5.0	17/04/2018	Description of Use Cases Updated	Marcus Hardt / KIT Valentin Kozlov / KIT Lara Lloret / CSIC
V6.0	15/04/2018	Technical requirements	Valentin Kozlov / KIT
V7.0	22/04/2018	Update on Research Communities Change in document structure	Lara Lloret / CSIC
V8.0	23/04/2018	External Review	Ignacio Blanquer / UPV
V9.0	24/04/2018	External Review	Fernando Aguilar / CSIC
V10.0	25/04/2018	Internal Review	Valentin Kozlov / KIT
V11.0	25/04/2018	Update on Requirements table	Valentin Kozlov / KIT Ignacio Heredia / CSIC Lara Lloret / CSIC
V12.0	26/04/2018	Internal Review	Álvaro López / CSIC

V13.0	27/04/2018	Update on Use Cases and Requirements	Wolfgang zu Castell / HMGU Werner Dubitzky / HMGU
V14.0	27/04/2018	Final version	Wolfgang zu Castell / HMGU Werner Dubitzky / HMGU Valentin Kozlov / KIT Ignacio Heredia / CSIC Lara Lloret / CSIC

Table of Contents

Executive summary

This report summarises the work of the WP2 on the *initial plan on the selection of Use Cases and the strategy for the DevOps pipelines* providing a key input to the design of the DEEP-HybridDataCloud (from now on just DEEP) testbed and how the DEEP solutions will be used.

A list of the Use Cases has been proposed and analysed in detail by the different research communities on the basis of the interest of the DEEP solutions, as shown in the following table:

Research Community	Use Case / Application
Deep Learning	Retinopathy detection
	Plant Classification with Deep Learning
	Deep Learning application for monitoring through satellite imagery
Massive Online Data Streams	Intrusion Detection
	Network Security Issues
	Anomaly Detection
Post-processing of massive amounts of data	Quantum Chromodynamics (Lattice QCD)

Table 1: Research communities and the use cases.

A complete list of requirements per Use Case has been compiled internally. The information has been collected by the Research Communities involved working hand in hand with the different experts in DEEP solutions in order to decide which is the best approach both to fulfil the Use Case requirements and exploit to the fullest the tools provided by the project. The list of requirements is the main outcome of this deliverable.

When considering the Use Cases, it was important to include the different roles (basic user, intermediate user, advanced user) in their analysis and try to find general solutions to support common requirements. These use cases correspond to the three defined user interaction categories described in this document.

A list of numerous requirements, classified by type (Computational/Storage/PaaS) and rank (critical/major/optional) has been used in turn to extract a short list of common requirements, classified into three categories:

- Computational Requirements
- Requirements linked to Storage
- Requirements on Infrastructure

The list with the final requirements is enumerated below. Further details are provided in the Common Requirements section in this deliverable. A more comprehensive table, including further information is provided as an appendix to this document.

#REQ	Description
CO#1	Access to GPUs with adequate performance for Deep Learning
CO#2	Support for multi-GPU computing nodes
CO#3	Sufficient amount of RAM (e.g. 1GB / core, or 64-128GB /node)
CO#4	Fast network connection to operate large (1-10TB) data
CO#5	Low latency interconnect
SO#1	Data caching for fast access
SO#2	Data storage for development (e.g. 1-10TB / Use Case)
SO#3	Persistent data storage for deployment (e.g. 1TB / Use Case)
PL#1	Containerization of Use Case applications
PL#2	Access restriction to a subset of authorised people
PL#3	FaaS / Function as a service
PL#4	Mobile applications (iOS, Android)
PL#5	Modularity
PL#6	Access restriction type

Table 2: List of common requirements.

The information collected from each of the use cases is provided as appendices to this document, and is stored online, since further interactions are expected during the lifetime of the project. This will be reported in “D2.2 – Intermediate Status Report about Use Cases” in month 18 of the project.

1. Introduction

Machine learning 'as-a-service' can clearly be seen as one of the main user requirements being asked for when thinking about using large-scale computing infrastructure. With ever more data being available, the wish to exploit this data is omnipresent. While machine learning pipelines for small-scale data sets are available in the form of several software libraries, large-scale learning tasks provide another level of challenge. With the need for a proper design of the learning task at hand, additional tasks result in the need to organize large-scale data, the provision of necessary computing power and storage capacity and, since large-scale data is commonly distributed, the orchestration of various infrastructure components at different places. It is obvious that such learning tasks cannot be managed by a user with domain knowledge in the field of application, only. Therefore, support by the infrastructure layer must break down the complexity of the task and allow the user to focus on what she/he is skilled on, i.e., modelling of the problem, evaluating and interpreting the results of the machine learning algorithms.

As a consequence, infrastructure providers have to understand the needs of their user communities and help them to combine their services in a way that encapsulates technical details the end user does not have to deal with. The goal of the DEEP project is to develop such applications along the lines of selected Use Cases. The latter will serve as blue prints for three classes of machine learning problems, which are prototypical for data intensive computing challenges. Note that the various steps of the machine learning process have different computing requirements. In particular, training a deep neural network typically requires a large amount of GPU resources, while execution of a trained network can be performed on a standard hardware. Similarly, some machine learning approaches rely on large storage being available, while others deal with data in a one-by-one fashion.

a) Deep Learning has turned out being a powerful tool to extract patterns from large datasets, allowing to identify complex patterns which are hard to extract with conventional algorithms. The power of Deep Learning arises from the availability of large collections of data being accessible through the Internet and advanced computing power, in particular GPU resources becoming more and more common. In its classical form, convolutional neural networks (CNN) are ideally suited to be trained on a GPU-based infrastructure.

b) Another challenge for data analysis results from machine learning on large-scale data streams. While initially, many algorithms have been designed assuming that the full dataset is available at the execution time of the learning algorithm, large scale data streams prohibit such an approach. Thus, algorithms both have to be able to use part of the data at a single step, only, and not hinder the flow of data through execution of training.

c) A third challenge results from large-scale data being produced by HPC simulations. Making such data accessible for machine learning and data analysis first requires to extract the data from the HPC resources in a suitable form and make it available for machine learning, which typically runs on a completely different architecture.

The computing pilots have been carefully selected to deal with these three challenges. The overall concept is to learn to understand the requirements of the user communities by working in detail on concrete Use Cases representing the prototypical challenges.

Note that the different Use Cases rely on varying levels of knowledge in machine learning technology on the side of the user communities. The users can therefore be classified in three categories depending on their machine learning knowledge and the use they want to make of the platform:

- The most **basic users** just want to use the tools that have been already developed and apply them to their data. They therefore need almost no machine learning knowledge. For example, they can take an already trained network for plant classification that has been containerised, and use it to classify their own plant images.
- The **intermediate users** want to retrain the available tools to perform the same task but fine tuning them to their own data. They still might not need high level knowledge on modelling of machine learning problems, but typically do need basic programming skills to prepare their own data into the appropriate format. Nevertheless, they can re-use the knowledge being captured in a trained network and adjust the network to their problem at hand by re-training the network on their own dataset. An example could be a user who takes the plant classification tool but retrains it to perform animal classification. From the point of view of infrastructure providers, this variant also relies on stronger computing infrastructure in order to be able to execute re-training.
- The **advanced users** are the ones that will develop their own machine learning tools. They therefore need to be competent in machine learning. This would be the case for example if we provided an image classification tool but the users wanted to perform object localisation, which is a fundamentally different task. They will then need to design their own neural network architecture, potentially re-using parts of the code from other tools. To these users we will offer an environment with the latest deep learning frameworks optimally supporting the development of new machine learning tools by releasing the user from dealing with the technical requirements of resource allocation and operation of the underlying hardware. Furthermore, once a tool has been developed, researchers can choose to easily share their tool with the community. In this case the infrastructure is asked to provide scalable resources as being demanded during while executing the training part.

All three levels will be addressed successively, starting with the straight forward ones.

2. Research Communities

Three Research Communities (Deep Learning, Massive Online Data Streams and Post-processing of massive amounts of data) represented by 4 different partners are participating in the WP2/NA2 in DEEP. The areas covered by the communities are: Biological and Medical Science, Computing Security, Physical Sciences, Citizen Science and Earth Observation. Table 3 provides the relation of Research Communities and partners, including the contacts that are directly contributing to DEEP

WP2. The numbering for the partners corresponds to the ID assigned in the Description of Work document.

#	Partner	Research Community	Area/Type	Contact
P0	CSIC	Deep Learning (Plant classification)	Citizen Science	Ignacio Heredia
		Massive Online Data Streams (Security issue)	Computing Security	Álvaro López
		Post-processing of massive amounts of data (Quantum Chromodynamics)	Physical Sciences	Isabel Campos
		Deep Learning (Satellite images)	Earth Observation	Daniel García
P7	CESNET	Massive Online Data Streams (Intrusion detection)	Computing Security	Tom Rebok
P8	IISAS	Massive Online Data Streams (Anomaly detection)	Computing Security	Giang Nguyen
P10	HMGU	Deep Learning (Retinopathy detection)	Biological and Medical Science	Wolfgang zu Castell

Table 3: Relation between research communities and partners.

The following information about the Research Communities is a summary of the complete information that has been provided directly by these partners through the documents included in the Appendix. It is compiled here as considered useful in particular for internal communication.

2.1. Biological and Medical Science

Deep learning approaches to biomedical image analysis have opened new opportunities in how diseases are diagnosed and treated. However, one drawback of automated analysis of medical images with deep learning is the requirement for sophisticated IT infrastructures (hardware, software, network). In this project, we will explore various ways to apply deep learning to analyse images in the context of retinopathy [Yau2012].

Retinopathy is a fast-growing cause of blindness worldwide, over 400 million people at risk from diabetic retinopathy alone. The disease can be successfully treated if it is detected early. Colour fundus retinal photography uses a fundus camera (a specialized low power microscope with an attached camera) to record colour images of the condition of the interior surface of the eye, in order to document the presence of disorders and monitor their changes over time. Specialized medical experts interpret such images and are thus able to detect the presence and stage of retinal eye diseases such as diabetic retinopathy. However, due to a lack of suitably qualified medical specialists in many parts of the world, comprehensive detection and treatment of the disease, particularly among underserved populations, is difficult. In this we facilitate and promote the application of automated deep learning classification of retinopathy based on colour fundus retinal photography images in three scenarios [Eul2017]: (a) Deployment and remote use of deep learning image classification. (b) Development of a new deep learning model based on a large integrated set

of colour fundus retinal photography images. (c) Same as (b) but based on a geographically distributed pool of such images.

Deep learning approaches in medical research pose an additional challenge to infrastructure providers since due to data privacy regulations, data can not automatically be assumed to be collected at one particular place. In the other hand, deep learning requires large amounts of data being available in order to deliver useful results. A cloud infrastructure is particularly well designed to deal with this challenge, since data can be moved to other computing infrastructures as well as code can. Therefore, training can be actually executed successively at different places, allowing the data owner to leave the training data at a particular site. Obviously, for such applications to be accepted by the community of biomedical researchers, the challenges of dealing with distributed IT infrastructures have to be fully encapsulated such that the user does not need to bother with these issues. Therefore, services being developed within DEEP will provide a solution to an otherwise hard to solve problem in medical image analysis.

2.2. Citizen Science

The potential of applying deep learning techniques for plant classification and its usage for citizen science in large-scale biodiversity monitoring has been discussed in recent publications [Heredia 2017]. Using near state-of-the-art convolutional network architectures, like ResNet50, achieves significant improvements in accuracy compared to the most widespread plant classification application in test sets composed of thousands of different species labels. The predictions can be confidently used as a baseline classification in citizen science communities like iNaturalist (or its Spanish fork, Natusfera) which in turn can share their data with biodiversity portals like the Global Biodiversity Information Facility (GBIF).

A citizen science approach is naturally suited to enlarge the set of labelled images needed for high quality classification of plant species. Allowing every interested citizen to use the network on his/her own images and to provide own suggestions for plant classification quickly increases the amount of labelled images which in turn allows to refine the classification after re-training of the network.

2.3. Earth Observation

The application of NN to pattern recognition in satellite images, and their combination with other in-situ measurements, opens new possibilities in areas like Ecosystems and Biodiversity. The EU Copernicus Programme relies on a family of dedicated Earth Observation missions called the Sentinels. The data acquired from these missions are systematically downlinked and processed to operational user products, and made available under an open and free license [Copernicus]. The current Sentinel data offer includes Sentinel-1A&B, Sentinel-2A and Sentinel-3A, and the Data Distribution Service operated by ESA provides access to more than one million products for on-line user download. The volume of available Sentinels user products is projected to increase, assuming the current operations context, at a rate of about 10 PB per year once the Sentinel-1/2/3 and 5p mission will have reached their full operational capacity. In parallel, the Copernicus services

operate in six different thematic areas, providing services and information free of charge to users, like the Copernicus Land Monitoring Service, that provides geographical information on land cover and on variables related, for instance, to the vegetation state or the water cycle. It supports applications in a variety of domains such as spatial planning, forest management, water management, agriculture and food security, etc. Copernicus users are increasingly adopting cloud-type solutions for the development of commercial and scientific applications using Sentinel products.

The project will promote the application of NN in the analysis of images applied to topics like the contribution to the estimation of EBV (Essential Biodiversity Variables), or the monitoring of Water quality, where the IFCA team is already contributing at international level in initiatives like LifeWatch ESFRI.

2.4. Computing Security

The usage of specialised hardware in order to speed up packet capturing, pre-processing and classification for network analysis is becoming very popular in recent times. Intrusion detection, or deep packet inspection systems are highly demanded application frameworks by network security analysts. In this context, compute-intensive networking is a clear trend to process large amounts of data that are continuously produced and require to be processed with time-limited constraints. Applications span from internet security, to analysis of network events with clear impact to risk analysis or Financial Services. The common feature to all these applications is that it is needed to process a continuous flow of information and react promptly to the generated events. For instance, using a recent Linux Kernel, an intrusion detection system is now able to process about 200 Mbit/s of IP traffic for security analysis depending on the rule set implemented in the system. Processing traffic in the 1-10Gbit/s threshold implies already deploying a packet load balancer in the Linux Kernel to distribute the packets to different acquisition channels, and then run several daemon instances, each one processing one different channel. In a medium-to-large-size computing centre the experience is that one ends up with order 50 Million of events per month to analyse. This comes of course after some basic pre-processing, excluding trivial TCP/IP alerts that generate on their own millions of events per hour, one ends up with order 100 Million of events per month to analyse. The problem is not that the number of events is too big to process, but rather that it is difficult to find actual problems, anomalies and vulnerabilities between those millions of events. Additional intelligence is need into the automation process to have an intrusion detection system that can cope with data rates expected in modern switches. The application of Artificial Intelligence (AI) techniques (like machine learning, in particular, deep learning, supervised neural networks, genetic algorithms, etc.) to process the alerts generated by an IDS or the network traffic (once it is filtered) is a known technique [Silva 2004], although their usage has not yet been generalised. For example, one anomaly may trigger several different kinds of alerts (for example, an SSH scan, then a successful connection from that scan, wrong TCP timestamps, download from strange URLs and then IRC traffic). Therefore, one event is typically not enough to classify something as an intrusion, but the combination of all of them. Nowadays, all IDS rely on a security analyst that needs to look at the alerts and classify them correctly, but just having a look at more than one million events per

day makes no sense. Automatic (simple) classification can be done, but this may hide true positives. The project will foster the application of AI techniques for intrusion detection based on an online aggregate of event streams. Security analysts would be able to feed their captured network events into the system, getting only interesting events as a result.

2.5. Physical Sciences

Quantum Chromodynamics (QCD) is the theory that describes the interaction responsible for the confinement of quarks inside hadrons (the so-called strong interaction). It is formulated in terms of quarks and gluons, which we believe are the basic degrees of freedom that make up hadronic matter: this is, protons, neutrons, and in general any particle composed of quarks. QCD has been very successful in predicting phenomena at very High Energies. In such regimes, the theory can be investigated analytically by using perturbation theory. However, when it comes to study the properties and interactions of the hadronic world (at the scale of the energy of the proton), the perturbative methods fail. In this domain one needs a non-perturbative treatment of QCD. Lattice QCD provides then the only mathematically well-defined method for calculating from first principles, for example, the hadronic spectrum: e.g. the mass of the proton, or the masses of the quarks themselves. In general, numerical simulations are often considered a tool to extract qualitative results on the behaviour of complex systems. This is not the case in Lattice QCD, where the simulations produce results that are exact, on the given lattice, up to statistical errors: an infinite computer, would solve QCD exactly. In experimental particle physics hadrons are studied by colliding protons at high energy (at the LHC for example). Obtaining the theoretical estimation of those masses, or the decay rates of unstable particles composed of quarks, is fundamental to explain the experimental phenomena, where in many cases the theoretical incertitude dominates the error bars. A well-known example is the scale of energy from which on the quarks bound into hadrons, the so-called Lambda parameter, which is theoretically accessible only by Lattice QCD simulations. This analysis work is often done in the small-medium size clusters available to the researchers in their local Labs. When the size of the simulated data reaches the order of Terabytes, moving those data across the labs of the research team to perform the analysis work, becomes a real issue that challenges every day the ingenuity of Lattice QCD researchers in the continent. Main reasons being insufficient access to storage/computing resources. Also, the very limited access to high speed networks due to “last mile” problems implies that moving the configurations from the HPC system to their local cluster becomes a problem, often extremely time consuming to solve. The analysis process consists in running the analysis software over all the configurations produced during the Monte Carlo history, and which are stored on a HPC filesystem. The output of the analysis is typically rather small. Encapsulating the analysis software (standard tools for statistical analysis) in a container docker-style is a trivial task. Providing the way for such container to be run, seamlessly, and in such a way that it has access to those data stored in the HPC system, would be of enormous help for this community.

3. Methodology used

This section presents the methodology used to gather the requirements of the different Research Communities. The process started with the development of a common template described below (see the Appendix for details), including different sections to compile the information that was considered to be relevant for DEEP. The idea of a comprehensive template as well as the structure of this template was agreed in the F2F WP2 meeting in Bologna in January, and the different communities completed a first version of their annexes along February. Along the biweekly teleconference meetings, the discussion on the different sections of the Annexes helped to prepare updated versions. These improved versions were then used to extract the specific and common requirements (see next section). The details are provided below.

3.1. Design of a template for Use Cases

In order to capture the community requirements in an efficient and systematic way, a common template was designed and has been used by the partners to provide this information.

The design of the template followed after an initial discussion at the kick-off meeting in Bologna about the well-known “impedance mismatch” between researchers and ICT teams, and follows 29148-2011 - ISO/IEC/IEEE International Standard [IEEE29148-2011] .

The design of the template uses a Case Study oriented approach. A Case Study is an implementation of a research method involving *an up-close, in-depth, and detailed examination of a subject of study (the case), as well as its related contextual conditions.* When collecting requirements, communities were informed to focus on Use Cases that are representative both of the research challenge and complexity but also of the possibilities offered by DEEP solutions on it. The Case Study is based on a set of User Stories, i.e. how the researcher describes the steps to solve each part of the problem addressed. User Stories are the starting point of Use Cases, where they are transformed into a description using software engineering terms (like the actors, scenario, preconditions, etc.). Use Cases are useful to capture the requirements that will be handled by the DEEP infrastructure and service work packages

The template serves as a structured framework with guiding questions concerned by DEEP service work packages. The inquiries cover the following aspects:

- *Executive summary on the Case Study*. A community has been asked to give background information about research challenge, explaining the expectation for DEEP technology and potential impacts the study case would deliver.
- *Introduction to the research Case Study*. this section provides a presentation of the Case Study, community roles to be involved, key technological and e-Infrastructure requirements, and potential exploitations.
- *Technical description of the Case Study*. Questions in this section intend to drill down into technical details to understand the computational functionalities a community would need. In the last section, the community was asked to describe the Case Study from the research point of view that they are familiar with. In this section, they are step-by-step guided to

break the Case Study into User Stories and think in more detail, in term of Use Cases, how those User Stories could be achieved through computational functions, how those computational elements interact with each other via workflows, and what the deployment framework and scenarios could be.

- *Data Requirements*. This section is specially designed since DEEP is also a data-oriented project. The community is asked to describe their Data Management Plans, data & metadata model, and operations involved in data life cycle, e.g., data acquisition, data curation, data process, data analysis, data visualization, and data publication.
- *Infrastructure technical requirements*. While previous sections focus on Use Cases, functional requirements and data model, this section focuses on e-Infrastructure resources and computing capacities, e.g., CPU, storage, and networking. It also includes the inquiry questions about AAI.
- *Connection with INDIGO solutions*. This section is left open to include the ideas and feedback of technology providers, in particular the DEEP JRA work packages, to consider how the community requirements could be efficiently and effectively supported by DEEP implementation.
- *Formal list of user requirements*. This section tries to put the focus on the preparation of detailed Use Cases starting from User Stories most relevant to the Case Study considered. A community is asked to list use stories based on data collection, curation, processing, analysis, simulation, etc. that are considered most relevant for the Case Study being analysed. In particular they are requested to include if possible an example of support for Big Data driven workflows for e-Science, with requirements for scientific workflows management, under a “Workflow as a Service” model, where the proper workflow engines will be selected according to user needs and requirements.

As described above, by including all these aspects of the Case Study, like computational functionalities, data model, technology in use, and capacity to request, the design ensures that filling such template will gather sufficient information for the DEEP implementations. The template has been reviewed and cleaned up internally by the WP2/NA2 team.

The template has also been reviewed and commented by DEEP JRA leaders to assure that it captures their expectations on the collection of requirements. The template, as well as each of the Annexes, is considered to be a live document along the whole life of the project.

3.2. Collection of input from pilot Use Cases

After obtaining the approvals within WP2 and from other WP leaders, the template was distributed to the Research Communities, asking them to provide at least a relevant Case Study. A brief summary of the context of these communities has been given in Section 3. This resulting collection of our Use Cases from several communities and scientific disciplines provides a relevant sample of demanding requirements for the DEEP Learning as a Service Platform (WP6). This information, including this deliverable, is public and can be shared with other technology providers that want to gain insights of communities’ needs.

3.3. Classification of pilot Use Cases according to their requirements from a machine learning perspective

The classical machine learning cycle starts with a set of data which is split into training data and test data. After training an appropriately chosen model, the test data is used to estimate the quality of the model with respect to its learning tasks. The latter can typically be related to either classification or regression. After the model has been trained and its accuracy being evaluated, the readily trained model can be used in applications for classification/regression. Depending on whether labelled data is available right at the beginning, learning algorithms can be divided into supervised and unsupervised methods.

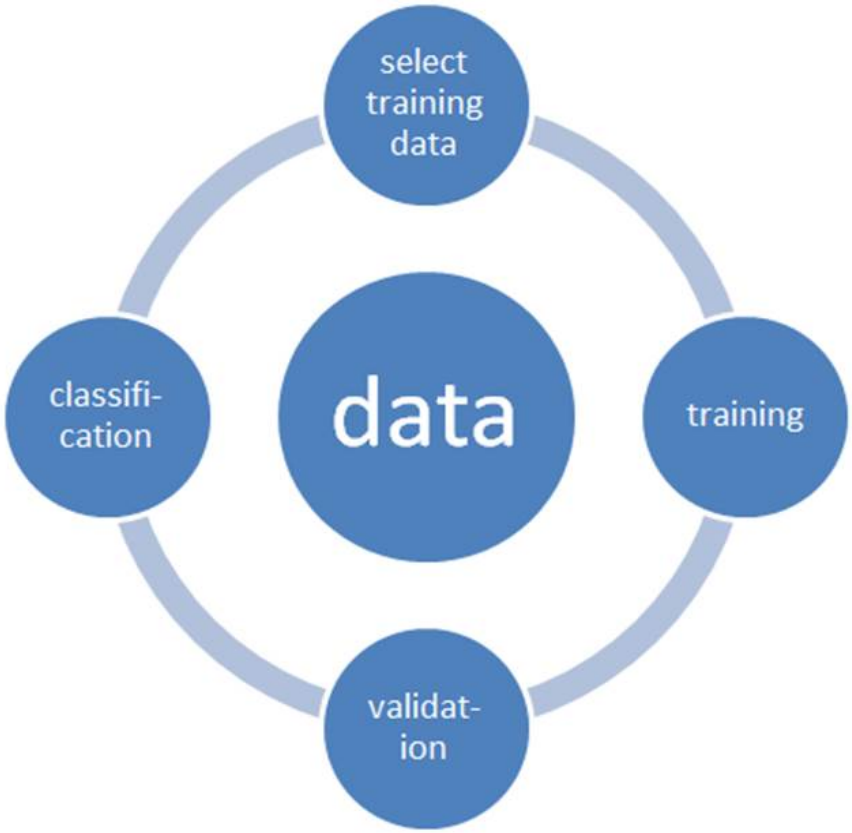

Figure 1: Learning cycle for a machine learning application

From a computing perspective the various steps of the learning cycle (Figure 1) are not all equal. While training is commonly computing intensive and relying on large amounts of data being available, validation and classification are normally much cheaper. Furthermore, the cycle is typically executed several times. An extreme case of moving through the cycle very often is given by online-learning algorithms. Similarly, data requirements for the various steps within the cycle are different. While some algorithms need large amount of data being available right from the beginning at the training phase, others work through the data in patches. Finally, depending on the learning algorithm being used, various ways of distributed computing and computing architecture are being needed.

For a first assessment of the pilot Use Cases, three categories of learning problems have been defined.

Category 1

The simplest category from the infrastructure point of view is given in terms of a readily trained model. Thus, some user has designed, trained and validated a model on his/her own dataset and now aims to provide the model as a tool for other users to be applied to their own datasets. In order to facilitate usage of the model, the developer casts the model into a ready-to-use container which can then be executed at another site using other data.

DEEP services will be developed to allow the end user to apply the model without the necessity to dig into technical details such as infrastructure being used, software libraries being installed and programming being done.

For dealing with category 1 learning tasks, no particular knowledge on either IT technology nor machine learning expertise is assumed to be provided by the user. The user rather represents a typical scientists with domain knowledge from his/her field of expertise.

Examples for this category are the Plant Classification network and the retinopathy network.

Category 2

Every trained machine learning model is capturing information on the data it has been trained on. This information can normally not be extracted from the model. It rather is part of the parametrization of the model as a whole. For example, a classical convolutional network architecture for a deep learning network for image analysis has several layers decomposing the image into artificial bits of information, leading to a very high-dimensional representation of the input image. A following set of layers then reduces the dimension, i.e. essentially solving a feature extraction task. Finally, the extracted features are then recombined in a third stack of layers in order to solve the classification task at hand, i.e. identify a cat in an image. Since deep learning networks need to be trained on large datasets, computational costs for training are commonly high. The essentials taken from the dataset are then represented in the weights of the convolutional neural network.

If a deep neural network has been trained on a large dataset for solving a particular classification problem, the network can also be modified to solve another classification problem. Towards this aim, one would keep the weights of the deeper layers of the network untouched and only re-train the network on a new dataset re-computing the weights of the outer layers of the network. Thus, the information being captured from the first learning phase leading to a suitable set of features derived from images is being kept. Only the part of the network needs to be re-trained, which uses the set of extracted features and recombines them to solve a new classification task.

DEEP services will allow the user to obtain a trained network, reset some of the model's parameters and re-train the model on a new dataset. Thus, the user must be able to access suitable hardware for executing the re-raining. The DEEP architecture model allows for such reverse movement in the deployment pipeline.

Learning tasks of category 2 assume the user to have at least some knowledge on machine learning next to the domain knowledge needed to model/understand the learning problem. But still no deeper knowledge of the underlying IT infrastructure is assumed to be provided.

Examples for category 2 are given by the retinopathy Use Case and the Satellite Imagery. Also the Massive Online Data Stream Use Case and the Post Processing of QCD Simulations fall into this category.

Category 3

Cloud technology allows to address machine learning tasks in a new way and thus also push the field of machine learning applications. As mentioned in the introduction, an example is given by data privacy restrictions which might not allow the user to gather all data at one site. Similar restrictions follow if the datasets being used a simply too large such that data movement is prohibitive. To nevertheless allow machine learning algorithms to be used, code need to be moved from site to site, executing training sequentially in a distributed way.

Thus, category 3 subsumes learning approaches which give the modeller full freedom over the architecture being used and allowing to provide specifications for data usage. DEEP services should aid the user in developing new tools with the cloud technology taking care of resource allocation, orchestration and deployment.

Clearly, learning tasks of this category can not be performed without any knowledge on the infrastructures being used, since the user is assumed to provide information on how data is distributed and can be accessed and also technical restrictions being inherent at the various data sites being involved.

Potential to exploit this category are given by the retinopathy Use Case and the Massive Online Data Streams Use Case.

The inherent increase in difficulty also leads to a natural order of working on the categories. Starting with readily available trained models of category 1 will allow to evaluate standard DEEP services with no special requirements. In a next step, category 2 problems will be addressed, increasing the requirements for the infrastructure providing WPs. Category 3 will finally be tackled once services provided by DEEP have reached their level of highest maturity.

3.4. Identification of technology requirements and prioritisation

The annexes provided by the user communities are very much Use Case focused. As described above, the user communities have been asked to report on their needs from a researcher perspective, starting from the topics in which they are more familiar (e.g. their research field, their software tools), rather than generic requirements for cloud, computing or storage technologies. The research Use Cases are translated into the requirements for the DEEP JRA work packages starting from the gaps identified in the User Stories detailed in the annexes.

Figure 2: Process of identification of the requirements from the User Stories.

The process that led to the final set of requirements is as described in Figure 2. The Use Case is the description of the workflow that the user needs to perform to do their research, the information include for example which type of data, where the dataset are hosted, which processing is performed on the input data and/or to generate the outputs and where the outputs have to be stored. The current state of the art is the way the workflow is currently implemented by the researchers, as reported in the annexes this includes a very diverse set of use of IT resources that goes from downloading locally the data to process in the personal computer and use its computing power, to using a computing cluster or even distributed e-infrastructures.

From the analysis of the current state of the art it has been possible to identify the constraints and the expected gains for the users, where they saw limitations in the work they are doing, and where chances of improvements are. The users either see limits in their current workflows or they need to increase their productivity, for example to analyse or produce bigger datasets, reduce waiting time, share their data or more in general collaborate with other researchers in a easier way. The specific constrains and limitations reported by the user communities have been grouped under a set of technical gaps, still linked to the user communities Use Cases but more focused on the actual technology used or potentially supporting the Use Case. These technical gaps or functional requirements are then translated in technical requirements that can be addressed by the DEEP solutions.

The constraints, gaps, functional requirements, technical requirements and the proposed solutions have been summarised in tables to be easily presented to the rest of the DEEP community. These tables are reported in the appendix. Given the considerable number of requirements, WP2 produced a first attempt of prioritisation.

During the first analysis of the annexes the WP2 team - based on the description of the Use Cases - classified the requirements in three levels of priority ranking: Critical, Major, and Optional. Where “Critical” means that without this development the Use Case cannot consistently improve (“must”), “Major” is a requirement that would further improve the workflow for the users, but it is less critical (“should”), while “Optional” means that the requirements is a nice-to-have but not critical (“can”). The Research Communities have then been asked to review the priorities and comment or modify them.

The prioritisation is also done by looking at the complete picture of all the requirements produced by the communities. Every Use Case provided by the user communities identified several technical requirements, and the DEEP solutions often fulfil several requirements, this already provided a first analysis of the importance of the developments planned in DEEP, some solutions will support more Use Cases and therefore can be more relevant. This information will be used as an input in the

development. The contacts of the user communities have been asked to check and comment or edit the summary tables to confirm that the analysis reflects their actual status.

It is important to mention that the requirements don't arise only from the Use Case needs. A requirement can be transparent for the Use Cases but can present a certain level of priority based on how critic it is for the development of the infrastructure (e.g. integration with the European Open Science Cloud, improvements in the overall infrastructure, etc...)

3.5. Interaction with JRA developers

The first step for the requirements gathered from WP2 is the DEEP Steering Committee (SC), where all the work packages are represented. The DEEP technical architecture, the services and the components that will be offered to the project users, will be defined within the SC and the requirements are the first input for this design process, since the communities are the main stakeholders of DEEP. This communication is not, of course, a one-way channel. The development packages have to analyse the requirements as they have been reported by WP2, identify and propose the solutions. To reach the optimum level of understanding of the Use Cases and the technical implications behind them, direct interaction between the developers groups in DEEP and the communities is foreseen following Agile approach [AgileDev].

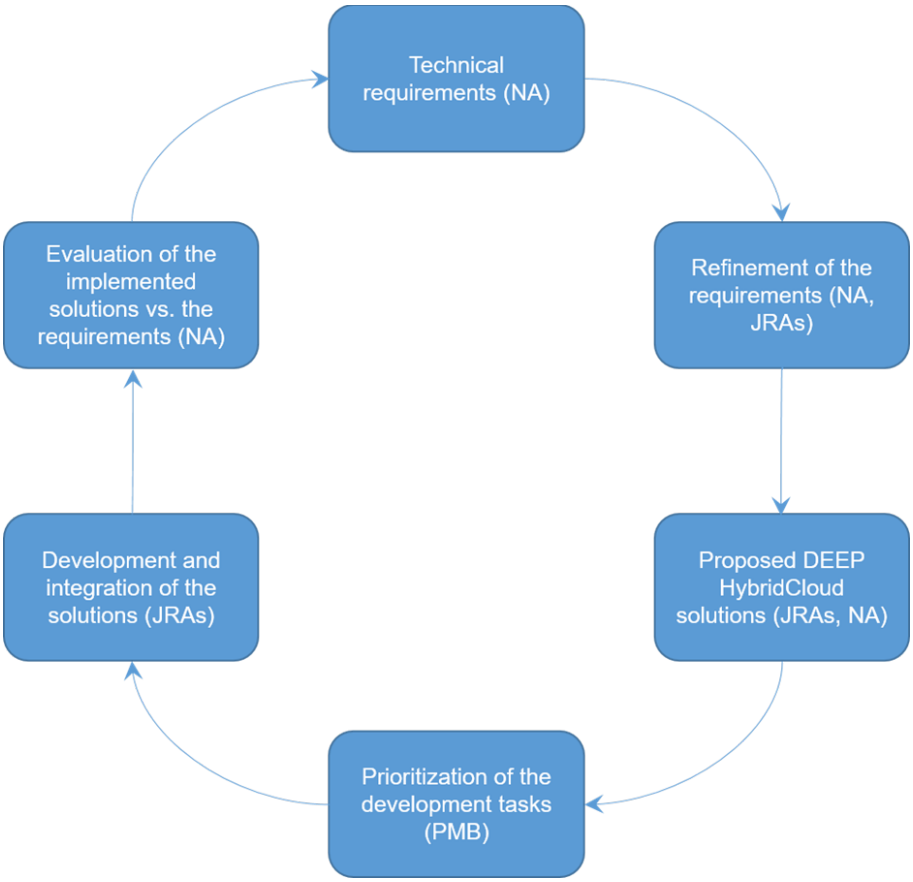

Figure 3: Requirement definition and implementation cycle

As the Figure 3 shows, we foresee a cyclic approach to requirements analysis. The beginning of the cycle is the first box where the technical requirements are defined within WP2. The requirements are then refined and analysed with the JRAs work packages teams, to reach a set of proposed solutions for development. The development and integration tasks will be then prioritised by the DEEP SC. Using as inputs the results of the development/integration activities further feedback is provided in form of Technical Requirements, as described in Section 5.3 Requirement tracking.

3.6. Incorporating input from external communities

The DEEP concept is attractive to many Use Cases that require HPC in general and GPUs in particular. Therefore, additional Use Cases have already expressed their interest in collaboration before the initial runs and setup of hardware has been possible. The external use-cases will be reviewed and prioritised in the DEEP Steering Committee, and engaged in the project through the relevant boards. These external communities will benefit from the examples given by the internal use-cases, that are naturally more evolved and more experienced in using the DEEP infrastructures.

4. Summary of Use Cases

This section provides an overview about the Use Cases proposed by the Research Communities. At the end of each Use Case the User Stories are included. They can be seen as macro-requirements that outline interactions and non-functional requirements. Additionally, the Annexes provide the complete input on the different Use Cases.

4.1. Deep Learning

The Case Study of Deep Learning is composed of three different Use Cases that all rely on classic Deep Learning and image recognition.

4.1.1. Use Case 1: Plant Classification with Deep Learning

This Use Case describes a tool to automatically identify plant species from images using Deep Learning [Heredia2017]. This can be very helpful to automatically monitor biodiversity at a large-scale and therefore relieving scientist from the tedious task of having to hand-label images, in a world where the amount of data to classify is far greater than the number of experts able to classify it.

In addition this tool can be easily retrained to perform image classification on a different datasets by a user without expert (machine learning) knowledge by performing the so called transfer learning.

This Use Case leverages the fact that everybody is equipped with a smartphone device. It wants to support collaborative citizen science platforms that have sprung and are now enabling users to easily share their observations and have them classified by experts. This case intends to collect those freely available observations and high quality labels to build Deep Learning tools to around them.

The scientific goals of this Use Case are the following:

- Produce a tool that is able to classify plants species from images.
- Have the results produced by the developed tools validated by biodiversity experts.
- Deploy this tool to automated monitoring of biodiversity. This means that we will make the tool available to be queried through an API so that communities and scientists can automatically retrieve predictions for each new observation they have. This means to shift the paradigm from a scenario where a few expert scientist are able to classify only a small amount of all the images generated by the community due to their limited amount of time, to a scenario where a machine will be able to instantly classify an unlimited amount of images. Therefore gathering (and classifying) the images taken from users in portals like iNaturalist [iNaturalist] we can have a real time picture of the state ecosystems in any part of the world.

Use Case	Plant classification with Deep Learning
Area	Citizen Science
Software and services used	Python, web services that will enable users to access the tools
Machine / Deep Learning tools	• Python: Numpy, Scipy. • Image processing: OpenCV, PIL. • DL: start with TensorFlow + Keras, in future may go for pyTorch
Computing	Multi-GPU nodes (GPUs satisfying for Deep Learning)
Memory requirements	At least 8 GB (GPU)
Networking	• From user local storage to cloud storage high enough to transfer user's own dataset. • Between a cloud storage and GPU as fast as possible, as it is a typical bottleneck.
Storage requirements (permanent, temporal)	• In the order of 10 TB (dataset for training)
External data access requirements	• Portals like iNaturalist (https://www.inaturalist.org) or Natusfera (http://natusfera.gbif.es) through an official API • PlantNet (https://identify.plantnet-project.org) via web scraping
Privacy	No privacy concerns as data are publicly available
Other requirements	
Other comments	• This Use Case will have millions of images so training will need to be distributed. • Going to run/develop the Use Case in a local machine, later will need containerized solution.
Relevant references or URLs	Heredia, I., 2017, May. Large-scale plant classification with deep neural networks. In *Proceedings of the Computing Frontiers Conference* (pp. 259-262). ACM.; arXiv:1706.03736

Table 4: Summary table for the use case "Plant Classification with Deep Learning".

The type of User Stories on the plant classification Use Case focus on easy re-usability. This Use Case is mainly intended for scientist that want to train and deploy an image classification service without any machine learning expertise. These User Stories include:

US1.1 - A zoologist has an image database of animals and wants to train an existing image classifier to automatically classify any new image. He wants to be able to upload his database to the platform taking into account his database is in the order of millions of images and therefore needs data storage in the order of TBs.

US1.2 - He wants to run a retraining on his database in a reasonable amount of time (days) and be able to keep track of the training status.

US1.3 - He wants to be able to easily deploy the newly trained model to make predictions on new images.

US1.4 - He wants to be able to share his new developed tool (the code and the training weights and parameters) with the community.

US1.5 - He wants other users to be able to build tools on top of his new tool.

US1.6 - A user has several images of plants and wants to identify them on his local computer. He wants an plug-and-play solution with a friendly interface.

US1.7 - A user has several images of plants and wants to identify them using a web service or a smartphone app. He wants this service to be fast and accessible through a friendly interface.

4.1.2. Use Case 2: Deep Learning application for monitoring through satellite imagery

With the latest missions launched by e.g. the European Space Agency (ESA), such as Sentinel, equipped with the latest technologies in multi-spectral sensors, we face an unprecedented amount of data with spatial and temporal resolutions never reached before. Exploring the potential of this data with state-of-the-art AI techniques like Deep Learning, could potentially change the way we think about and protect our planet's resources.

Possible applications of machine learning techniques range from remote object detection, terrain segmentation to meteorological prediction [Jean2016][Albert 2017][Zhang2017]. As a first approach, this use case will implement a tool to reliably detect water from Sentinel multi-spectral bands, thus demonstrating the potential of combining satellite imagery and machine learning techniques in a cloud infrastructure. In addition to its usefulness (for automatically monitoring water resources without expensive on-site devices), this application will set the foundations to build other tools with satellite imagery, from object segmentation to forecasting.

The scientific goals are

- Enable monitors for ecosystems and the design of better policies regarding the environment.
- Enable the capabilities of the European biodiversity platforms such as LifeWatch.

- Develop a detection and prediction system that combines the latest Deep Learning techniques with satellite data. An environment where you can process and analyse different satellite maps, choosing the place, the date, etc.

Use Case	Deep Learning application for monitoring through satellite imagery
Area	Earth Observation
Software and services used	Python, orchestration of containers (Mesos, Kubernetes?), data orchestration, web services that will enable users to access the tools
Machine / Deep Learning tools	• Python: Numpy, Scipy, Pandas, matplotlib, os; snap-api, sentinel-sat, earthengine-api. • Image processing: OpenCV and PIL. • DL: start with Tensorflow + Keras, in future may go for pyTorch
Computing	CPUs and powerful GPUs for Deep Learning (8 GB GPU memory preferred)
Memory requirements	64-128 GB
Networking	As fast as possible, as data transfer rate between a cloud storage and GPU is the main bottleneck
Storage requirements (permanent, temporal)	Feasible approach: • 100s of GBs as permanent storage (intermediate representations, the code, and the neural network) • 10s of TBs as temporary storage (raw data)
External data access requirements	• ESA: https://scihub.copernicus.eu/dhus • NASA: https://earthexplorer.usgs.gov/ accessed via a corresponding API to copy data locally
Privacy	No privacy concerns as the datasets (Sentinel) are open access
Other requirements	
Other comments	• A node with single GPU is probably good enough to start with. • Going to run/develop the Use Case in a local machine, later will need containerised solution.
Relevant references or URLs	• https://scihub.copernicus.eu/userguide/ • https://sentinel.esa.int/documents/247904/690755/Sentinel_Data_Legal_Notice

Table 5: Summary table for the use case "Deep Learning application for monitoring through satellite imagery".

The type of User Stories on the satellite imagery Use Case focus on developing an easy pipeline to work with satellite data. This pipeline will be partially reusable but is no longer a plug-and-play application like in the plants classification Use Case. These User Stories include:

US2.1 - A government wants to reuse the developed model to create an application able to automatically detect fires from satellite images. They want to be able to upload their own satellite imagery database or choose to use available module to retrieve satellite imagery.

US2.2 - They want to run retraining on a reasonable amount of time (days) and be able to keep track of the training status.

US2.3 - They want to be able to easily deploy the newly trained model to make predictions on new images.

US2.4 - They want to be able to share his new developed tool (the code and the training weights and parameters) with the community.

US2.5 - They want other users to be able to build tools on top of their new tool.

US2.6 - A conservation agency wants to develop a novel application to be able to count the number of trees in a national park. They will therefore need to access an environment supporting all major deep learning frameworks to develop their tool (in the case this tool doesn't exist already).

US2.7 - A user wants to find water spots in a region. He wants to be able to easily submit queries through a friendly interface (allowing inputs as region coordinates, dates...) and have the results returned quickly.

4.1.3. Use Case 3: Retinopathy detection

Retinopathy is a fast-growing cause of blindness worldwide, over 400 million people at risk from diabetic retinopathy alone [Yau2012]. The disease can be successfully treated if it is detected early. Colour fundus retinal photography uses a fundus camera (a specialised low power microscope with an attached camera) to record colour images of the condition of the interior surface of the eye, in order to document the presence of disorders and monitor their change over time. Specialised medical experts interpret such images and are able to detect the presence and stage of retinal eye disease such as diabetic retinopathy. However, due to a lack of suitably qualified medical specialists in many parts of the world a comprehensive detection and treatment of the disease is difficult. This use case focuses on a deep learning approach to automated classification of retinopathy based on colour fundus retinal photography images [Eul2017].

The scientific goals of use case are

- To develop and evaluate a tool facilitating the classification (based on deep learning) of retinopathy stage and progression based on digital colour fundus retinal photography images.
- To improve automated classification retinopathy stage (Healthy, Mild, Medium, Severe) and reconstruct disease progression by means of deep learning.
- To address the need for a comprehensive and automated method for DR screening.

Use Case	Deep learning application for classification of disease progression of diabetic retinopathy.
Area	Biological and medical sciences
Software and services used	Python, orchestration of containers, data orchestration, web services that will enable users to access the tools.
Machine / Deep Learning tools	• Python: Numpy, scipy, pandas, matplotlib, os • Image processing: OpenCV and PIL • Deep learning: Sart with Tensorflow and Keras
Computing	CPUs and powerful GPUs for Deep Learning (8 GB GPU memory preferred)
Memory requirements	64-128 GB
Networking	As fast as possible, as data transfer rate between a cloud storage and GPU is the main bottleneck for the learning phase in Deep Learning. A decent network performance may also be required in the distributed Deep Learning sub-use case where the training data resides in different locations.
Storage requirements (permanent, temporal)	Feasible approach: • 100s of GBs as temporary storage (for intermediate representations, the code, and the neural network) • 100s of GBs as permanent storage (raw data including raw image data).
External data access requirements	Kaggle: https://www.kaggle.com/c/diabetic-retinopathy-detection
Privacy	Sub-use case 3 explores Deep Learning model construction on inherently distributed training data. One reason why such a scenario might be important is privacy-preserving model construction.
Other requirements	Currently none. Additional requirements may arise during the development, deployment and evaluation of the use case.
Other comments	A node with single GPU is sufficient to start with.
Relevant references or URL	• https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html • https://www.kaggle.com/c/diabetic-retinopathy-detection

Table 6: Summary table for the use case "Retinopathy detection".

Here we distinguish three increasingly complex user stories involving domain users, data scientist and modellers, and a consortium of users with various needs.

> **US3.1** – A domain practitioner/expert user, possibly located in region where large-scale computing infrastructures are not readily available, needs to interpret medical images (e.g. colour fundus retinal photography images) obtained from a local screening program. A deep learning classifier deployed on the DEEP infrastructure is used to assist the user with the interpretation (e.g. classification) task.

US3.2 – A machine learning expert or medical image analyst seeks to develop a deep learning image classifier from an image data set. Lacking a suitable IT infrastructure (software, GPU, storage) to develop such a model, the expert user uses the DEEP infrastructure for all stages of model development and deployment.

US3.3 – A collection of image datasets adhering to the same modality and format is generated in various locations. For technical, regulatory or other reasons, it is infeasible to physically combine these data sets into a larger integrated data pool. To capitalise on the potential of such a data pool for constructing a deep learning classification model, the data set owners use the DEEP infrastructure to learn a deep learning classifier "incrementally". In such an approach, the training phase of deep learning is organised in such a way that intermediate versions of the model are moved from data location to data location to construct the overall model without having to move the local data sets. Various approaches to realise this are conceivable.

4.2. Computing security

4.2.1. Use Case 4: Massive Online Data Streams

A data stream is a sequence of data items that become available over time. This type of data is common to certain environments, such as sensor networks, web logs, etc. Stream data is quite different from static data sets. In particular, stream data is potentially infinite, may be fast flowing, and may require a fast response. The Use Case is focused on hybrid detection that combines misuse detection in real-time with external intelligence modules built based on the heuristic analysis techniques using ML/DL techniques. The problem is not that the number of events is too big to process, but rather that it is difficult to find actual problems, anomalies and vulnerabilities between those millions of events. Additional intelligence is needed in the automation process to have an Intrusion Detection System (IDS) that can cope with data rates expected in modern switches.

Misuse (or signature) detection is a kind of reactive security solutions. It is already a part of IDS implementation in many modern infrastructure systems. The advantages of the solution is fast and effective. Its weak side, except for the reactiveness, is the lack of flexibility and inability to detect new threats e.g. zero-days attacks or polymorphic malware. Furthermore, having a look at more than one million events per day makes no sense. Heuristic analysis operates on the base of experience and can be either reactive or proactive [Dua 2016] [Tran 2017] [Ng 2018]. These techniques make attempts to generalize security issue characteristics that differentiate them from a normal state.

In our Use Case, the hybrid solution will be realised as follows. IDS for misuse detection will operate by adapting to external intelligent analyser modules, i.e. using ML/DL models produced based on security data analytics, intrusion, and anomaly detection. These intelligent modules will evaluate and return results based on probabilities to the e-infrastructure IDS software module. Then, the extended IDS can be considered as a technology complementary to the security data analytics, intrusion, anomaly detection. These last technologies, for the most part, work on a large number of historical events and try to extract suspicious behaviour i.e., sequences of events

containing suspicious activities. IDS, on the other hand, acts on live input streams and raises alarms when certain sequences are detected in near real-time. Our solution will combine advantages from both misuse and heuristic solutions i.e. to improve cyber-security resilience for e-infrastructure.

The scientific goals of this case include

- Definition of an architecture for data intake, analysis and storage.
- Research and development of different tools e.g. monitoring tools, decision support modules, prediction module with cloud supports that allow data analysis.
- Research and development of intelligent modules using ML/DL to analyse and to get meaningful insights of massive online data. Applying and testing different approaches toward cyber-security aspects focused on event detection.
- Accelerate Use Case development using cloud e-infrastructure modern technology advantages such as isolated independent environment, that provide built-in security, portability, and flexibility features.

Use Case	Massive Online Data Streams
Area	Computing Security
Software used	• Python: NumPy, SciPy, Pandas + an extendable environment for Python development with standard packages (Matplotlib, pydot, h5py, etc.), possibly with the support of Jupyter notebooks. • Other tools: Snort IDS, Sguil, barnyard
Machine / Deep Learning tools	Keras, TensorFlow, Scikit-Learn, (optional) DL4J, (optional) VW
Computing	CPU and GPU
Memory requirements	High RAM capacity
Networking	Accessibility of the data from both CPUs and GPUs with low latency in the communication between nodes.
Storage requirements (permanent, temporal)	1-10TB permanent
External data access requirements	Access to the data should be restricted to a subset of authorised people with access log archive and monitoring.
Privacy	Private data, in-site due to the sensitiveness of the collected raw data from the security viewpoint
Other requirements	• High internal caching to reduce the number of I/O operations • Authentication and authorisation service is required. • Architecture needed for analysing data online and archive historical data for reuse. Lambda architecture with usage of Apache Spark/Flink/Kafka for stream processing can be a good candidate.
Other comments	
Relevant references or URLs	• https://www.sciencedirect.com/science/article/pii/S1877050917306865

	• https://www.sciencedirect.com/science/article/pii/S0169023X17303063?via%3Dihub

Table 7: Summary table for the use case "Massive Online Data Streams".

- **US4.1:** A user wants to deploy a customized Big Data architecture (including data-intake tools, data analysis and storage)
- **US4.2:** A user wants the possibility to redefine the architecture of the solution as needed
- **US4.3:** A user wants to take advantage of a ready to use solution to be deployed on their data centre
- **US4.4:** A user wants to retrain online and offline an existing anomaly detection model using a different dataset (for example adding a new IDS to the system)
- **US4.5:** Endpoints and services should be secured, with proper authentication and authorization methods.

4.3. Post-Processing of Large amounts of data

4.3.1. Use Case 5: Post processing of QCD simulations

Quantum Chromodynamics (QCD) is the theory that describes the interaction responsible for the confinement of quarks inside hadrons, the so-called *strong interaction.* Investigating the properties of QCD requires different techniques depending on the scale of energy we are interested in. In particular at low energies, at the scale of the proton mass where quark confinement takes place, we must rely on non-perturbative methods, notably Lattice QCD simulations.

Competitive Lattice QCD simulations spread nowadays over thousands of cores in HPC systems with low latency interconnects [openQCD]. The amount of data that needs to be analysed in a medium-size project is on the order of the Terabyte.

Data analysis presents technical challenges to the researchers involved. Those challenges can be traced down to the lack of a flexible environment to move data across the network and analyse them in a flexible way. The design of a computing environment in which the researcher can, pipeline the analysis of Lattice configurations would greatly improve the efficiency of the process of configuration analysis.

The purpose of this Use Case in DEEP is to serve as pilot for designing such a data configuration tool, in such a way as to have general applicability for similar usage scenarios in other scientific areas.

Use Case	Post processing of QCD simulations
Area	Physical Sciences
Software used	OpenQCD
Machine / Deep Learning tools	No
Computing	HPC systems with low-latency interconnects

Memory requirements	Depending on the case: from 1GB to 4GB / core
Networking	Fast access to handle downloading of data from the HPC system to the analysis facility
Storage requirements (permanent, temporal)	For the showcasing of the Use Case, a maximum of 1TB is needed.
External data access requirements	The data are stored in a HPC system, and need to be accessed from outside.
Privacy	Private to collaboration members
Other requirements	The code is highly optimized for machines with current Intel or AMD processors, but will run correctly on any system that complies with the ISO C89 (formerly ANSI C) and the MPI 1.2 standards. For the purpose of testing and code development, the programs can also be run on a desktop or laptop computer. All what is needed for this is a compliant C compiler and a local MPI installation such as Open MPI.
Other comments	
Relevant references or URLs	http://luscher.web.cern.ch/luscher/openQCD

Table 8: Summary table for the use case "Post processing of QCD simulations".

US5.1 - A Lattice collaboration needs to analyse a large number of configurations. Usually every member of the collaboration involved in the analysis needs access to the data. Using a containerised version of the analysis code, the system should be able to automatise the retrieval of configurations from the HPC system to the cloud storage system, and run the analysis on cloud resources as well.

US5.2 - A Lattice collaboration wants to move the configurations to long term storage. For that a number of checksums and related metadata applying operations need to be done. A containerised version of this process should be able to automatise the checksum and metadata creation. Ideally those data could then by sent to a long term storage tape-like, also in an automated way.

5. Requirements

As described in section 4 on Methodology, the previous Use Cases are summarized in section 5 and described in detail in the corresponding Appendix were used directly to gather the list of requirements.

5.1. Requirements gathered from Use Cases

As an initial analysis, and not precluding a more detailed and direct analysis of the Annexes directly by JRA teams, a list of requirements per Case Study was compiled internally within WP2. These tables are available online in the project confluence wiki and as an appendix.

These tables were analysed by a WP2 team of integrators well aware of infrastructure components and the availability of solutions together with the use-case presenters. The result of this analysis have been collected in the form of requirements in one table per use-case. There we collected the Research Community, Requirement enumeration (#), description, priority rank (Critical/Major/Optional), current solution, gap, etc. The requirements are classified as "Critical", if they are felt as such in the input provided by the corresponding Research Community. This may mean that either the requirement is satisfied by a current solution and needs to be preserved, or that it must be implemented in the new solution. "Major" requirements indicate that they obviously will be implemented, but with lower priority than "Critical" ones. The complete list is provided in an Appendix of this same document for completeness.

5.2. Common Requirements

As a next step in the proposed analysis, a new list was prepared within WP2 trying to identify in the previous table the Common Requirements. This required a certain degree of interpretation and consolidation of the given requirements. These consolidated requirements are classified into three categories:

- CO: Computational requirements
- SO: Requirements linked to Storage
- PL: Requirements related to Service

They are also classified according to their rank (c = critical, m = major, o = optional) assigned on the basis of the Use Cases requirements. The result is shown in the following table:

#REQ	Description	Type	Rank	Proposed improvement
CO#1	Access to GPUs with adequate performance for Deep Learning	Computing	c	Testbed nodes equipped with GPUs
CO#2	Support for multi-GPU computing nodes	Computing	o	Wherever possible equip testbed nodes with multi-GPUs
CO#3	Sufficient amount of RAM (e.g. 1GB / core, or 64-128GB /node)	Computing/ RAM	c	Provide enough RAM in testbed nodes
CO#4	Fast network connection to operate large (1-10TB) data	Network	c	
CO#5	Low latency interconnect	Network	m	Clusters with low latency interconnects
SO#1	Data caching for fast access	Storage	m	Each node has to have fast access to the data in order to feed them to the GPU
SO#2	Data storage for development (e.g. 1-10TB / Use Case)	Storage	m	

SO#3	Persistent data storage for deployment (e.g. 1TB / Use Case)	Storage	o	Enough storage capacity in testbeds
PL#1	Containerization of Use Case applications	Service	c	Provide support for Use Cases to containerize their applications
PL#2	Access restriction to a subset of authorised people	AAI	c	Plants / Satellite: if needed or requested can easily integrate with OIDC
PL#3	FaaS / Function as a service	Service	m	Serverless architecture for providing the plant recognition service. An API will query this serverless architecture and return predictions to users
PL#4	Mobile applications (iOS, Android)	Service	m	Would use FaaS in Plants Inference Use Case
PL#5	Modularity	Service	m	Enable the composition of different modules to deploy complex architectures and topologies
PL#6	Access restriction type	AAI	o	AAI should be compliant with AARC recommendations and the AARC blueprint

Table 9: Consolidated requirements of use cases.

This list of requirements is the main outcome of this deliverable, and it is provided as input to the work of the JRA teams within DEEP project.

5.3. Requirement tracking

In order to keep track of the user stories, the technical requirements and the involved developments to be carried out by the project JRAs, WP2 and WP3 have defined a set of processes based on the collaborative tools used within the DEEP-HybridDataCloud project [DEEP-requirement-manager], Confluence and JIRA. Confluence is being used as the main source of information for the project, holding all the working documents for the project. JIRA is the issue tracker where the development activities are managed and tracked. JIRA and Confluence are tightly linked, facilitating the information flow between the two tools.

The defined processes involve the definition of *product blueprints* (containing specific requirements) that are linked to individual *user stories* using the Confluence tool. These user stories are then linked to JIRA *Story issues*, where the development takes place. JIRA allows the breakdown of these issues into individual working items (subsequent *JIRA issues*), depending on the complexity of the tasks to be performed. By using these processes, evolving user needs and requirements can be incorporated into the development cycle of the project at any stage. Consolidated reports will be generated accordingly.

6. Plan for Use Cases and next steps

The main activity in WP2 is to adequately support the use-cases, therefore the following approach was chosen: A WP2 team of integrators will work as an interface between the infrastructure and platform providers (WP3) and the case-studies. With their know how on the cloud infrastructure, they will teach and enable the DevOps concept [Bass 2015] with the use-case representatives. A concrete example comes from the use of containers (e.g. [Gomes 2017]). The integrators understand the workflows and tools for creating containers. Based on consolidated use-case requirements this enables them to provide template containers to the use-case representatives. The containers can be further extended to use-case specific requirements, e.g. to include optimization, own installations and corresponding dependencies (e.g. specific Deep Learning code, libraries etc.). Once this is done, the integrators will provide support and contacts for running the containers on the provided infrastructures. This will include help in data access, scaling of the solution and verification of the results.

WP2 has identified a three step approach for three working phases of the project.

6.1.1. Phase 1 (PM1-12): Development infrastructures

In the initial phase WP2 team will understand details on the use of the infrastructures and establish the network with the use-case representatives. This will lay the base for the DevOps approach that we follow in the DEEP project. The immediate next steps for this are:

- Test runs on the infrastructures to investigate the support for GPU and Infiniband.
- Provision of template containers to the use-case representatives so they can start the first steps in their DevOps approach
- Deployment of PoC pilots on the infrastructure (such as Jupyter notebooks) that can also be used for development by the case-studies.

6.1.2. Phase 2 (PM12-18): Inference workloads

Building on the first phase, more complex tasks can be addressed. From the requirement analysis it was learned that the most common task is inference. This has slightly more complex requirements on the infrastructure and the workflows, because it implies that multiple components be started with dependencies on each other. In addition to the initial pilots, this will require access to the pre-trained datasets. This adds the requirement to manage bundles of containers with the dependency of multiple components.

In addition, there will be at this point investigation of the support for serverless solutions, that provide function as a service use-cases interfaces to the infrastructure. These interfaces are ideal for supporting mobile applications and of course easily deployable web services.

Fully (or partially) retrain / and validate models on DEEP on data set of up to 100,000 images.

6.1.3. Phase 3 (PM18-30): Distributed training

The experience with the first two phases will put us into a very good starting position to address training of deep neural networks. These workloads are in principle similar to those from inference, but they require much larger volumes of data, network bandwidth and processing power, and therefore should significantly profit from being distributed across multiple computing nodes.

The following table details the different tasks within the WP2 and the initial plan to accomplish the WP milestones.

		2017	2018	2019	2020
		11 12	1 2 3 4 5 6 7 8 9 10 11 12	1 2 3 4 5 6 7 8 9 10 11 12	1 2 3 4
Task 2.1	Prototype description of the pilot showcases	1 2	3 4 5 6 7 8 9 10 11 12	13 14 15 16 17 18 19 20 21 22 23 24 25 26	27 28 29 30
	Initial collection of requirements from research communities				
	Analysis of Use Cases and identification of common requirements				
Task 2.2	Test & Validation on testbeds	1 2	3 4 5 6 7 8 9 10 11 12	13 14 15 16 17 18 19 20 21 22 23 24 25 26	27 28 29 30
	Test runs on the infrastructures to investigate the support for GPU and Infiniband				
	Provision of initial template containers to the use-case representatives				
	Implementation of DevOps pipeline				
	Adaptation of pilot showcases to the available testbed resources				
Task 2.3	Deployment and testing at large scale	1 2	3 4 5 6 7 8 9 10 11 12	13 14 15 16 17 18 19 20 21 22 23 24 25 26	27 28 29 30
	Measure the scalability of the services in homogeneous and hybrid infrastructures				
	Produce a report detailing the scaling tests performed				
Milestones and deliverables					
D2.1	Initial Plan on the selection of use cases and the strategy for the DevOps pipelines				
D2.2	Intermediate Status Report about use case integration and DevOps methods				
D2.3	Final Report about use case integration				
MS2.1	Technology Assessment and initial selection of use-cases, M3, verified by D2.1				
MS2.2	Integration of use-cases in testbed completed, DevOps approach operational				
MS2.3	Final report after testing at large scale				

Table 10: Initial implementation time plan.

7. References and Links

[AgileDev] Agile software development, In *Wikipedia*. Retrieved April 2018, from https://en.wikipedia.org/wiki/Agile_software_development

[Albert 2017] A. Albert and M. Gonzalez. Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. *Proceedings of the ACM KDD*

[Bass 2015] Bass, Len; Weber, Ingo; Zhu, Liming. *DevOps: A Software Architect's Perspective. ISBN 978-0134049847.*

[DEEPUseCasesGDoc] Requirements for DEEP https://docs.google.com/spreadsheets/d/1RgtbkKTOnxsodFjksLro_j999AYD-YJ55d-pl0a8H5Q/edit#gid=1671433853

[Eul2017] Eulenberg, P. et al. (2017). Reconstructing cell cycle and disease progression using deep learning. *Nature Communications* 8:463.

[Buggenthin 2017] Buggenthin, F., Buettner, F., Hoppe, P. S., Endele, M., Kroiss, M., Strasser, M., ... & Schroeder, T. (2017). Prospective identification of hematopoietic lineage choice by deep learning. *Nature methods*, *14*(4), 403.

[Rasmus 2015] Rasmus, A., Berglund, M., Honkala, M., Valpola, H., & Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems* (pp. 3546-3554).

[Copernicus] http://www.copernicus.e u/main/data-access

[DEEP-requirement-manager] https://confluence.deep-hybrid-datacloud.eu/x/KARv

[Dua 2016] Dua, S. and Du, X., 2016. *Data mining and machine learning in cybersecurity*. CRC press.

[Gomes 2017] Gomes, J., Campos, I.: Researchers Advance User-Level Container Solution for HPC, https://www.hpcwire.com/2017/12/18/researchers-advance-user-level-container-solution-hpc/, accessed Mar 2018

[Heredia 2017] Ignacio Heredia, 2017. Large-Scale Plant Classification with Deep Neural Networks. *Proceedings of the Computing Frontiers Conference* , 259-262.

[IEEE29148-2011] 29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering. Persistent Link: https://ieeexplore.ieee.org/servlet/opac?punumber=6146377

[Jean 2016] N. Jean et al, 2016. Combining satellite imagery and machine learning to predict poverty, *Science*, Vol. 353, Issue 6301, pp. 790-794

[iNaturalist] https://www.inaturalist.org/

[Ng 2018] Nguyen, G., Nguyen, B.M., Tran, D. and Hluchy, L., 2018. A heuristics approach to mine behavioural data logs in mobile malware detection system. *Data & Knowledge Engineering*.

[openQCD] Simulation program for lattice QCD, http://luscher.web.cern.ch/luscher/openQCD/

[Tran 2017] Tran, D., Tran, N., Nguyen, G. and Nguyen, B.M., 2017. A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness. *Procedia Computer Science*, *108*, pp.365-374.

[Yau2012] Yau, J.W., Rogers, S.L., Kawasaki, R., Lamoureux, E.L., Kowalski, J.W., et al. (2012) Global prevalence and major risk factors of diabetic retinopathy. Diabetes Care 35: 556–564.

[Zhang 2017] Zhang, A., Liu, X., Gros, A., & Tiecke, T. (2017). Building Detection from Satellite Images on a Global Scale. *arXiv preprint arXiv:1707.08952*.

8. Glossary

AAI	Authentication and Authorization Infrastructure
AI	Artificial Intelligence
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
DEEP	Short for DEEP-HybridDataCloud
DL	Deep Learning
DM	Data Mining
DNN	Deep Neural Network
EBV	Essential Biodiversity Variables
ESA	European Space Agency
ESFRI	European Strategy Forum on Research Infrastructures
GBIF	Global Biodiversity Information Facility
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HPC	High-Performance Computing
IDS	Intrusion Detection System
IRC	Internet Relay Chat
IT	Information Technology
JRA	Joint Research Activities
LHC	Large Hadron Collider
ML	Machine Learning
MPI	Message Passing Interface
NA	Networking Activities
NN	Neural Network
PMB	Project Management Board
QCD	Quantum ChromoDynamics
REST	REpresentational State Transfer

RNN	Recurrent Neural Network
TCP	Transmission Control Protocol

9. List of figures

10. List of tables

11. Appendix – Requirements gathered from Use Cases

Use case	Req #	Comment – Request	Requirement	Type[1]	Rank (C / M / O)	Current workflow/solution	User Story
Plants	PLT#1		10TB	Storage	m	Local disk Storage might be remote, training will be able to download data in batches	US1.1
	PLT#2		GPUs (multi GPU nodes, if possible) GPU specs: NVIDIA GPU with at least 8GB of memory	Computing	c		US1.2 US1.7
Post Processing	PP#1		1TB for the proof of concept	Storage	m		US5.1 US5.2
	PP#2		Clusters with low latency interconnects	Computing	m		US5.1 US5.2
	PP#3		1GB / core	RAM/core	c		US5.1 US5.2
	PP#4		Fast access to handle downloading from HPC mainframe to the cloud facility	Networking	m		US5.1 US5.2
	PP#5		Data Restricted to collaboration members	Privacy	m		

1 Computing / Storage / PaaS service

Use case	Req #	Comment – Request	Requirement	Type	Rank (C / M / O)	Current workflow/solution	User Story
MODS (development)	MODSdev#1		1-10TB	Storage	m	For reduced and cleaned data in training (development phase). The data amount grows with time and will be stored as data pool for training. Expected amount of compressed monitoring data/logs is 1TB per site per year.	US4.1
	MODSdev#2	multiple nodes with one GPU each can be enough for performance	GPUs, the number of nodes (if possible) on demands	Computing	c	the availability of multiple nodes will shorten training in development by data parallelism in certain training intervals	US4.4
	MODSdev#3		1 multi-GPU node (if possible)	Computing	o	a multi-GPU node can be interesting for optional tests (e.g. speed-up, between-card communications) in comparison with single-GPU node. The use can be limited for a certain interval i.e. the Use Case will not block computation resource for very long time.	US4.4
	MODSdev#4	a minimum is 8GB GPU memory	Higher is better	GPU Mem	m		US4.1
	MODSdev#5		Faster is better	Networking	m		US4.1 US4.4
			access restricted to a subset of authorised people	Privacy	c		US4.5

Use case	Req #	Comment – Request	Requirement	Type	Rank (C / M / O)	Current workflow/solution	User Story
MODS (deployment)	MODSdep#1		1 GPU per 1 site	Computing	c	containerization deployment (e.g. docker/udocker) of several modules: ELK, IDS, ML/DL. The aim of deployment in production is to be fast and effective.	US4.3
	MODSdep#2	a minimum is 8GB GPU memory	Higher is better	GPU Mem	m		US4.1
	MODSdep#3		Faster is better	Networking	c	if the storage node is separate from computing nodes, then the long distance among them can be a problem from the network latency viewpoint	US4.4
	MODSdep#4		in-situ for production, access restricted to a subset of authorised people in the functionality testbed	Privacy	c		US4.5
Satellite Data	SD#1		1TB temporary data Storage (for the training process)	Storage	m		US2.1
	SD#2		1 TB persistent data Storage	Storage	o		US2.1
	SD#3		GPUs (multi GPU nodes, if possible) GPU specs: NVIDIA GPU with at least 8GB of memory	Computing	c		US2.2 US2.7
	SD#4		64 - 128 GB RAM	Computing/ RAM	c		US2.2

Use case	Req #	Comment – Request	Requirement	Type	Rank (C / M / O)	Current workflow/solution	User Story
Retinopathy detection	RD#1		100s of GBs as temporary storage	Storage	c	Features from multiple intermediate CNN layers may generate 100 GBs of temporary data. Current handling of this data is ad hoc.	US3.1 US3.2
	RD#2		100s of GBs as permanent storage	Storage	m	The raw image data is in the order of 100 GB. Current data handling is ad hoc.	US3.2
	RD#3	A node with single GPU is sufficient to start with.	GPUs. GPU specs: NVIDIA GPU with at least 8GB of memory	Computing	c	Multiple GPUs could potentially speed up CNN training.	US3.2
	RD#4		64-128 GB	Computing/ RAM	m		US3.1 US3.2
	RD#5		As fast as possible network between a cloud storage and GPU	Networking	m		US3.2
	RD#6		CNN model construction (training) based in inherently distributed data, i.e. data that has not been physically integrated. Privacy or other regulatory for technical reasons may require such a mode of learning.	Privacy	o	Currently, this is not done but there are potential applications in biomedical, financial, etc. settings.	US3.3

Use case	Req #	Comment – Request	Requirement	Type	Rank (C / M / O)	Current workflow/solution	User Story
Shared Requirements	Shar#1		Containerization	Service	c		US1.3 US1.6 US2.3 US2.6 US5.1 US5.2
	Shar#2		Mobile apps for inference	Service	m		US1.7
	Shar#3		Modularity	Platform	m		US1.5 US2.5 US3.2 US4.2
	Shar#4		FaaS / Function as a service	Computing	m	Deploy a serverless architecture for providing access to the predictions from the tools. An API will query this serverless architecture and return the predictions to the user.	US1.3 US2.3 US3.1
	Shar#5		Model sharing	Service	o		US1.4 US2.4 US3.1 US4.3
	Shar#6		Access Control will be implemented in REST interface if required		o		US1.7 US2.7

Table A1: Requirements derived from Case Studies.

12. Appendix – Use Cases description

The complete Use Cases description and documents can be found online, as an appendix to this document, in the following url: http://hdl.handle.net/10261/164311