ATMOSPHERE

Adaptive, Trustworthy, Manageable, Orchestrated, Secure Privacy-assuring Hybrid, Ecosystem for REsilient Cloud Computing

REQUIREMENT ANALYSIS

DELIVERABLE D7.1

Grant Agreement number:	H2020-EUB-2017/EUB-01-2017-GA: 777154
Start Date of Project:	01.11.2017
Duration of Project:	24 months
Work Package:	WP7 – Pilot and use cases
Due Date:	31/03/2018
Submission Date:	01/04/2018
Partner Responsible:	UPV
Dissemination Level:	PU (Public)
Nature:	R (Report)
Author(s):	Ignacio Blanquer (UPV)
Reviewer(s):	

RNP

Change Log

Version	Date	Description	Author(s)
v1.0	12/02/2018	Table of contents and complete initial version	Ignacio Blanquer (UPV), Amanda Calatrava (UPV)
v2.0	1/03/2018	General use cases	Ignacio Blanquer (UPV), Antonio Pinho (UFMG), Angel Alberich (QUIBIM)
v3.0	3/03/2018	Architecture	Ignacio Blanquer (UPV)
v4.0	11/03/2018	User stories	Ignacio Blanquer (UPV)
v5.0	12/03/2018	Analysis of the User Stories requirements by WP.	Ignacio Blanquer (UPV)
v6.0	20/3/2018	Description of the Use case	Conrado Calvo (UPV), Antonio Pinho (UFMG)
v7.0	26/03/2018	Introduction, conclusions, executive summary	Ignacio Blanquer (UPV)
v8.0	28/03/2018	Final version for review	Ignacio Blanquer (UPV), Antonio Pinho (UFMG), Fabio Castro (QUIBIM), George Teodoro (UnB), Ángel Alberich (QUIBIM)

Document Review

Review	Version	Date	Reviewers	Comments

Table of Contents

INTRODUCTION **6**

Scope of the document 6

Target Audience 6

Structure of the document. 6

PROCEDURE FOR THE ANALYSIS OF REQUIREMENTS **7**

ATMOSPHERE USE CASE OVERVIEW **8**

ATMOSPHERE roles 11

ATMOSPHERE sub-cases 12

Application developers 12

Application managers 13

Final end-users 14

Trustworthiness properties 15

USE CASE PRELIMINARY DESIGN **17**

USER STORIES **18**

Basic user stories 18

User stories with respect to the trustworthiness properties 22

Security and privacy use stories 22

Stability and Performance use stories 24

Fairness and Transparency use stories 25

REQUIREMENTS **26**

CONCLUSIONS **33**

ACRONYMS AND ABBREVIATIONS **34**

Disclaimer

Adaptive, Trustworthy, Manageable, Orchestrated, Secure, Privacy-assuring, Hybrid Ecosystem for REsilient Cloud Computing (2017-2019) (hereinafter "ATMOSPHERE") is a Research and Innovation Action funded by the European Commission under the H2020 Programme, Call identifier: *H2020-EUB-2017*, grant agreement No 777154, topic: *EUB-1-2017 Cloud computing, including security aspects;* and the Secretary of Politics of Informatics (SEPIN) of the Brazilian Ministry of Science and Technology (MCTI) under the corresponding matching Brazilian Call for proposals: *4º Chamada Coordenada Programa de Cooperação Brasil-União Europeia em Tecnologias da Informação e Comunicação – TIC*.

This document contains information on core activities, findings, and outcomes of ATMOSPHERE project. Any references to content in both website content and documents should clearly indicate the authors, source, organisation and date of publication.

The document has been produced with the co-funding of the European Commission and the Secretary of Politics of Informatics of Brazil. The content of this publication is the sole responsibility of the ATMOSPHERE Consortium and cannot be considered to reflect the views of the European Commission nor the Secretary of Politics of Informatics of Brazil.

Executive Summary

ATMOSPHERE aims at measuring and improving the trustworthiness of application and cloud services. The concept of trustworthiness is conceived in its multiple dimensions, addressing security, privacy, repeatability, stability, robustness, performance, fairness, explanation among others. The specific metrics and measures for trustworthiness are discussed in deliverable D3.1.

Although ATMOSPHERE focuses on developing the measures, models and actuators for improving trustworthiness, as well as providing a framework to run them, ATMOSPHERE will also implement a pilot use case to experiment and demonstrate this features. The pilot use case focuses on Rheumatic Heart Disease (RHD), a heart disease caused by poorly treating a set of infectious diseases that cause damages in the heart walls and valves. If detected, the infections can be easily managed, but if they remain undetected, it can cause severe morbidity. The project will use echocardiography images obtained from children in rural areas to automatically evaluate the risk of RHD according to the World Heart Organization parameters. The social impact of such application is very high, as this disease affects children and treatment is affordable and effective while damage if untreated can be definitive and even lead to casualties.

The pilot case involves processing medical data with different level of sensitivity. The project has substituted all identifiable critical data with fake data to avoid facing ethical issues but will use this data fields as samples for verifying the management of privacy and the identification of privacy risks. Therefore, the pilot case is a relevant example for properties such as privacy, reliability, robustness, predictable performance, fairness and explainability.

The development and usage of cloud services involve three actors (application developer, application manager and end-user). ATMOSPHERE has identified 31 user stories that describe the interactions of the three types of actors with the platform. These user stories involve actions such as the interaction with the infrastructure, the application delivery, the customisation of the trustworthiness properties, the definition of actuators to correct deviations on the trustworthiness properties, and the dynamic monitoring of applications. These 31 user stories led to identifying 14 requirements to drive the development of the ATMOSPHERE layers.

The pilot case is a cornerstone development to demonstrate and guide the functionality and properties of the ATMOSPHERE platform.

1. INTRODUCTION

ATMOSPHERE is a challenging collaborative project working in the area of trustworthiness cloud services. The main aim of the project is to provide means to measure and improve the trustworthiness of applications running on the cloud, considering trustworthiness in a broad sense. The project will produce a set of frameworks, services and components to evaluate properties such as reliability, performance, security, confidentiality, privacy management, stability, fairness and explanation in a federated cloud environment.

1.1. Scope of the document

The project includes a pilot use case to demonstrate the services and guide the development with a set of realistic user stories, use cases and requirements. The purpose of this document is to identify and analyse such user stories, use cases and requirements, through a systematic and collaborative procedure.

The document includes the procedure followed for the identification of requirements, the overview of the use case, focused on the remote analysis of echocardio images, the identification of user stories and requirements and their interpretation. The document also includes a preliminary design of the application and the conclusions.

We want to stress that the objective of this workpackage is not to develop a trustworthy cloud-based telemedicine service but to demonstrate the services developed in the frame of ATMOSPHERE and prove that they can be used to increase different dimensions of trustworthiness of the applications.

1.2. Target Audience

The document is mainly intended for internal use, although it is publicly released. The main target of this document is the global team of technical experts of the ATMOSPHERE project, including WP3, WP4, WP5 and WP6.

1.3. Structure of the document.

The document comprises eight sections. After this introduction, we define the procedure for the analysis of requirements. Then, the document introduces the use case, describing the aims, roles and the sub-cases. For this use case, we propose the preliminary design of the use case and extract the user stories that will be used for extracting the requirements. Then the requirements are presented, and the document ends up with the Conclusions and the acronyms sections.

2. PROCEDURE FOR THE ANALYSIS OF REQUIREMENTS

The analysis of requirements starts understanding the particularities of the user scenario from both the application developers and the end-users. This description is compiled in this section and it has been performed by the Use Case experts. Those experts elaborated a questionnaire that has been circulated among selected representative members, addressing the whole lifecycle of the use case. From the information in the questionnaire, user stories have been identified and requirements from them.

The action plan includes activities that would be completed in 4 phases (elicitation, analysis, specification and validation), shown in Table 1, along the project life cycle.

Activity	***Requires***	***Produces***
	Elicitation	
The expected user classes (actors) and other stakeholders are identified from the scenarios described by users		*Report on use case communities*
Users and developers reach a common understanding of the tasks and goals of the use case actors	*Report on use case communities, KOM Use case description*	*Discussion on telcos*
Users describe the environment in which they usually perform their work		*Inventory of applications and data sources*
User's quality expectations are discussed	*Inventory of applications and data sources*	*Discussion on telcos and e-mail*
	Analysis	
High-level requirements are described in the form of User Stories		*Initial list of Success Stories*
New functional requirements are generated from the analysis of the information	*Initial list of Success Stories*	*A complete list of requirements is defined*

Requirements are allocated to infrastructure functions and components	*A complete list of requirements*	*WP3, WP4, WP5, and WP6 advance in the definition of the software architecture*
Implementation priorities are identified	*As above*	*Requirements dependencies*
Specification		
Requirements are transformed into technical documents and diagrams.	*Final list of use case requirements*	*D7.1 – Requirement Analysis.*
The design of the pilot is revised and detailed	*Pilot application design*	*D7.2 Design of the Biomarker Pilot Application*
Validation		
Documented requirements are reviewed and accepted.	*D7.1 is approved*	
Early prototypes are developed and validated in the infrastructure.	*Integration tests are successfully passed*	*D7.3 Pilot demonstrator*
Users confirm that the use case implementation meets their needs.	*User acceptance tests are successfully passed*	*D7.6 – Validation of the Requirements*

Table 1 – Phases of the analysis of use case requirements

3. ATMOSPHERE USE CASE OVERVIEW

ATMOSPHERE focuses on measuring and improving the multiple dimensions of the trustworthiness. The ATMOSPHERE use case serves as a demonstrative case in an area in which most of the properties identified are relevant.

The use case area is the analysis of medical data related with cardiology. Cardiology data is collected from spread rural areas by trained technicians and later evaluated remotely by medical experts. The management of clinical data requires privacy protection, security, repeatability, high-availability, robustness and fairness. Moreover, the use case is defined in a

geographically distributed environment, also serving as a use case for the federated computing resources.

The objective of the use case is to process a massive set of medical images, along with additional metadata and clinical information, efficiently and securely, to extract features that could be used to assist and even automate diagnosis. More precisely, the use case focuses on the characterization of Echo-cardio images obtained in rural areas to obtain early surrogates for Rheumatic Heart Disease (RHD).

The RHD, which involves damage to one or more heart valves after acute rheumatic fever episodes, is a disease that can be easily treated in its early stages. RHD remains the main origin of heart valve disease in the developing world. However, it may produce severe damage to the heart if it remains untreated, including death. Affected heart valves may remain severely stretched and/or scarred with deteriorated mechanical function, and normal blood flow through damaged valves interrupted. Abnormal blood flow cycles may include either backward flow in case of stretched valves that do not close appropriately, or blocked diastolic flow in case of inadequate valve opening. This may lead to pathological mitral regurgitation, with retrograde flow impacting in the upper chamber walls further leading to myocardial remodeling and subsequent likelihood for developing the substrate for sustained arrhythmic events. In these scenarios, the heart function deteriorates quickly. Undiagnosed and untreated RHD may lead to heart failure and subsequent risk of fatal arrhythmias, stroke, endocarditis and other systemic complications causing progressive disability, reduction in the quality of life, and, increased likelihood to end in premature death on young adults. While heart surgery may prolong life managing some of these problems does not cure it, only early detection could help stopping its progression to remodeled stages. However, nowadays, early detection is limited since there is no single test to early diagnose accurately either acute rheumatic fever or early development of RHD.

Cardiac echocardiography has been used to assess valve mechanical function and appropriate flow interaction during the cardiac cycle. By analyzing the echo-cardio images, essential features could be extracted. Echo-cardio imaging techniques allow to determine key features related to several findings on pathological mitral regurgitation, pathologic aortic regurgitation, and morphological features of RHD in the mitral valve or the aortic valve, and/or mitral stenosis. In rural areas, it is recommended that significant subclinical valve lesions be labeled as probable RHD until proven otherwise. In 2001, a World Health Organization (WHO) Expert Committee established a consensus for the echocardiographic diagnosis of subclinical RHD based on the detection of valvular regurgitation by Doppler interrogation of the cardiac valves, grading the severity to allow distinction of physiological and pathological flow in diseased conditions. The World Heart Federation has defined a set of conditions based on echocardiographic features to classify patients into four categories: Definitive RHD, Borderline, Normal or Other abnormalities, based primarily on such echo-Doppler features of valve dysfunction. While the currently utilized World Health Organization (WHO) criteria for echocardiographic diagnosis of subclinical RHD emphasize the presence of pathological valve regurgitation do not include more precise additional morphological, mechanical or mechanical flow dynamics features of RHD for early detection of RHD, mainly when there is no clear indication of pathological regurgitation. Indeed, the boundary between physiological valve

regurgitation and authentic but minimal rheumatic lesions remains difficult to discern in some cases.

Therefore, developing new imaging markers or a combined stratification score based on significant features extracted from massive a set of images could aid on prospectively identifying patients at risk. Image post-processing and image classification techniques are required to obtain the features that drive such classification. These techniques will definitely lead to the individual processing of medical images and the subsequent training of classifying methods on subsets of the whole collection of images.

Doppler echocardiography will be carried as in previous studies in spread regions of the Brazilian area. Specifically, GE VividQ and VSCAN, and optionally Philips ultrasound systems will be used employing a cardiac sector transducer in the range of 100-300 micron resolution and frame rates in 2D up to 140-740frames/second. Output formats will include either '.AVI' movie files or DICOM package files. Doppler and 2D protocols will be used. Echocardiographic recordings will be carried by trained and experienced echo-cardiographers following a pre-stablished protocol and subsequently blinded to the feature extraction carried in the study. An expert physician will determine left ventricular volumes and ejection fraction (EF), as well as, left ventricular mass, following traditional supervised strategies from Simpson's method and Devereux formulae.

To assess the echocardiographic RHD prevalence across clinical and subclinical cases, determination of features derived from the consensus recommendations by WHO criteria will be also performed, only considering Doppler characteristics to evaluate valve structure and function dynamics. In this consensus, RHD is associated to a pathological valve regurgitation jet greater than 1cm in length with a peak velocity of 2.5m/s persisting throughout systole and diastole. Additionally, we will also explore further valve morphological features such as leaflet morphology and mobility (abnormal thickening and motion), and indications of stenosis. Traditional Doppler diastolic features will be determined and averaged across at least 3-10 consecutive beats: the early (E) and atrial filling (A) mitral valve peak velocities, E/A ratio, deceleration time of early filling and isovolumic contraction and relaxation time from transmitral flow, peak systolic (S'), early diastolic (E') and late diastolic (A') mitral annular myocardial velocity of the left ventricular septal and lateral walls, and E/E' as a surrogate index of LV filling pressure. Hemodynamic, systolic and morphological parameters will also be considered.

Two different post-processing strategies will be evaluated, one based on conventional model-driven image analysis algorithms that allow the extraction of features that are then fed into a classifier and another one directly based on a data-driven classifier that learns from the images extracting the features that better express the dataset characteristics by means of deep learning techniques, which in the case of imaging data, are based on Convolutional Neural Networks (CNN).

For the first approach, automated and unsupervised algorithms for image feature quantification will be tested. Simultaneous ECG acquisition will help to select appropriate windows for analysis of each feature. After optimization and removal of artifact reverberation, digital sequence filtering, motion tracking, elastic registration and unsupervised segmentation

based on a modified active contour algorithm the structures of interest will be unequivocally determined. Image post-processing will allow identifying traditional Doppler features. Packages optimized for high-performance medical image processing algorithms will be employed. Furthermore, data-driven deep learning features will be explored on the dataset. All derived features will be used to feed matching learning and advanced classification algorithms to build physiologically relevant models for evaluation of accuracy, goodness, strength of association to the endpoints of the study and power of discrimination.

In the second approach, artificial intelligence (AI) techniques, specifically deep learning techniques, will be applied to create a classifier for the screening of the echocardio images on this specific pathology (RHD). Convolutional Neural Networks (CNNs) will be developed and applied to achieve this objective, which can be divided in two steps:

1. The CNN classifier should serve as a first screening to differentiate between normal and abnormal echo-cardio studies.
2. After successfully building the normal/abnormal CNN classifier, create new classifiers to assess the actual pathological process or processes which caused the disease, e.g. abnormal valve functioning.

First, the annotated database will be supervised, cleaned and organized to ensure the quality of the data before developing the classifier. After the data has been properly prepared for its use, it will be split into training, validation and testing datasets. The training dataset will be used to create the model based on CNN, the validation dataset will be used to estimate the performance of the classifier during the training phase, and the testing dataset will be used to evaluate its accuracy with previously unseen images. Once the algorithms have been successfully validated, they will be integrated in the ATMOSPHERE technological framework to test it as a use case.

This large database of features will aid to discriminate strongly associated and independent predictors of RHD progression. Physiological conditions and constraints will be applied to build principle-based models as long as it is possible or, challenge the models accordingly. We hypothesize that adding different layers of features from massive dataset will help building robust markers, classifiers and clinically relevant scores for testing in prospective studies and increase case detection rates.

3.1. ATMOSPHERE roles

ATMOSPHERE identifies three roles relevant to the application level.

- Final end-users are data-scientists and medical imaging engineers who can use ATMOSPHERE platform to process Medical Imaging data using their applications and to leverage the services provided by ATMOSPHERE. We do not consider physicians at this level, as ATMOSPHERE aims at enhancing the trustworthiness of applications, but not at supporting data analysis in production.

- Application developers are software engineers who code and register applications on the ATMOSPHERE platform, benefiting from its evaluation capabilities to measure and improve the trustworthiness of their applications.
- Application managers, as the people in charge of deploying the applications on a cloud infrastructure and operating it.

Although this use case focuses on Medical Imaging, the ATMOSPHERE platform does not bound to the particularities of this use case, and it can be applied to many other data analytic problems.

3.2. ATMOSPHERE sub-cases

This section analyses the ATMOSPHERE use case to derive sub-cases for the user roles identified to be relevant to the application level. The sub-cases relate to simplified interactions that the different roles expect from the platform. The document does not include an analysis of how or where such requirements should be addressed. In some cases, the document presents examples that should not be taken as the desired implementation, but as examples for clarifying the requirement.

3.2.1. Application developers

Application developers interact with the platform to measure the trustworthiness of their applications and to increase it by using the ATMOSPHERE services. Application developers mainly require easing application development; increasing trustworthiness; and improving application delivery. Based on these expectancies, as an application developer, I need:

- A storage system to store persistent and temporal data (Mandatory). Applications require a storage system for files, relational databases and NoSQL databases that could be deployed efficiently on top of the federation.
- A job scheduler to execute parallel Spark jobs (Mandatory). The federation should provide the means to deploy a submission endpoint for this type of jobs that could fulfil the requirements defined in the trustworthy properties.
- A job scheduler to execute batch jobs (Mandatory). Equivalent to the previous use case, but referred to batch jobs, which could have particular dependencies.
- A way to describe the application and their dependencies (Mandatory). Applications are defined as a code in a declarative way. We propose to use Ansible roles, TOSCA documents and Dockerfiles to enable describing the different nodes and relations that compose an application. A coherent mechanism must be selected, so maintenance is minimal.
- A way to describe trustworthy properties (Mandatory). Trustworthy properties have many dimensions. There should be a coherent model in which an application can

define properties such as Quality of Service (e.g. an expected allocation of resources in a given time frame), privacy, fairness, and any other property.
- A platform to evaluate the trustworthiness of an application (Mandatory). ATMOSPHERE should provide an assessment both at design time and at runtime of the trustworthiness of an application.
- Automatic horizontal elasticity (Mandatory). The platform must allocate the adequate number of nodes to deal with the workload considering the trustworthiness features.
- A monitoring system with plugin capability for metrics, rules and actuators (Mandatory). A monitoring system to provide metrics of the trustworthy properties in runtime is needed. The monitoring system should enable defining plugins for application-defined metrics and actuators.

3.2.2. Application managers

Application managers deal with the deployment of an application and all its dependencies. An application manager needs credentials to access the federation and to requests resources on it.

As an application manager, I need:

- To register applications defined in a declarative way so they can be deployed in the federation (Mandatory). The applications developed in the project will be defined in a declarative language so they can be deployed on the fly. Recipes for automatically building up containers are also required. Descriptive documents should be compatible with OASIS TOSCA for higher compatibility. Reusability of the recipes is a must. Deployment may require the application manager to previously register a prebuilt container or Virtual Machine Image on a specific site of the federation.
- To adjust the application description to the trustworthy features (Mandatory). The application descriptions may have customisable parameters that can be updated according to a target based on the trustworthiness properties. For example, the number of resources, the allocation of GPGPUs, or other resources should be automatically adjusted in the application description if the desired deadline requires it.
- To deploy the applications on the federation (Mandatory). The applications that are correctly coded may involve several nodes or instances that must be cross-configured. The deployment should be considered as a whole, managing all the instances coordinately. The deployment may include requirements that must be fulfilled by the resources (such as the availability of GPGPUs or SGX support). This requirement includes additional considerations: The system should issue a clear error message if deployment cannot be fulfilled on a specific site or the whole federation; Access to unconfigured resources should be possible if the configuration fails, for debugging purposes; The system should provide complete information about the configuration process.

- To reconfigure the applications on the federation (Mandatory). It should be possible to change the configuration of a running application, either triggered by the application manager or automatically by an actuator. The application manager may consider including new software components, which should be applied on the application in the future. Also, the actuator may decide to add or to remove nodes to adjust the infrastructure automatically to the existing workload.
- To undeploy an application deployed on the federation (Mandatory). An application may comprise several resources which must be treated as whole to minimise orphan resources and to facilitate management.
- To obtain information about the applications deployed on the federation (Mandatory). The system must provide a list of available applications deployed by the application manager, as well as information about the resources and trustworthiness metrics of a given application.

3.2.3. Final end-users

A final user only needs to host the user credentials for accessing the application. He or she does not interact directly with the resources but with the endpoints provided by the application manager.

As an end user, I need:

- To store, query and download medical imaging data as files (Mandatory). Through the applications, the end users can upload medical images as files in a convenient way and retrieve it efficiently by the processing services.
- To store and query relational data in a database (Mandatory). Clinical data may be stored in a relational database, so this should be supported.
- To store and query unstructured data in a database (Mandatory). The outcome of the processing is likely to be coded as a structured report, which could be efficiently stored in NoSQL databases.
- To encrypt the data stored in a database, so it fulfils the data protection regulation constraints (Mandatory). Some clinical data may have privacy risks which could be minimised by encryption. Data should not be accessible even for system administrators unless they are provided with the access credentials.
- To encrypt the data stored in the filesystem, so it fulfils the data protection regulation constraints (Desirable). As the project only considers the imaging modality of echocardio, the risk of identification will be low. However, in a general scenario, this may not be the case.
- To entirely or partially anonymise the data (Mandatory). The requirement may impose a storage format to the user who should convert the data to this format.
- To execute a data processing algorithm through the applications provided by Application Managers (Mandatory) on the platform. The type of jobs supported depends on the application.

- To know the trustworthy scores for a given application, both at design time and during the execution (Mandatory). There are many trustworthiness scores relevant for the users, and there should be a way to query all of them.
- To tune up specific trustworthy parameters related to the application related to execution performance, privacy guarantees, fairness or explanation (Recommendable). Some of the trustworthiness scores may be customizable or even could require additional input from the users.

3.3. Trustworthiness properties

The use case requires a high level of trustworthiness, which should be assessed and provided by both the application and the platform. According to the trustworthiness properties defined in the DoA and which will be extensively described in D3.1, table 2 shows how those properties relate to the use case.

Property	Metric	score
Security	Vulnerability	Provide automatic verification of known the vulnerabilities of the use case source code.
	Test coverage	Use case source code should have coverage tests higher than a defined threshold.
Privacy Assurance	Risk Minimization	The application requires the execution in an encrypted environment when accessing private data or using IPR protected software.
	Privacy score	The application should have a privacy risk score propagated to the outcomes of the processing. For example, an application that extracts unique features from images and relates them to socioeconomic and demographic factors can produce data that could lead to re-identification. This property also requires data to be annotated concerning privacy.
	Sensitivity	The classification models should check the relevance of individual subjects in the model. The platform should provide the means to analyse how the classification model varies when removing individual cases or critical fields from training.
	Risk Minimization	Outlier detection of behaviours that could lead to identifying risky patterns even if no security break happens.
Stability	No degradation of performance over time	The use case application should not degrade memory or processing performance over time above a specified threshold.
	Reproducible	The use case application trustworthiness (performance and other

	deployment	trustworthiness scores) to be predictable when deployed in different infrastructures.
	The sensitivity of the algorithm	The use case application accuracy should not vary in different deployments (containers, VMs, a different number of resources, GPGPUs).
Isolation	Assessment of Isolation threats	The isolation of the processing applications is critical when processing personal data.
	Sustained trustworthiness in a competitive scenario	The performance to be equivalent to an isolated and competitive infrastructure (shared with other processes).
Fairness	Results are not discriminating based on gender or race	The platform should provide the means to evaluate the potential discrimination of the results based on gender or race (no other personal features are considered according to the nature of the data). This property can be achieved by analysing the results of the model separately for the different subgroups and considering all the subgroups. A significant difference may be reasonable when the problem has a direct correlation with sex or race.
Performance	Minimisation of runtime	If the application supports it, ATMOSPHERE should provide means to minimise execution runtime by using distributed resources or specific devices such as GPGPU. The procedure should be convenient for the user. For example, the user could select if runtime should me minimised (using his/her complete quota of resources and specific resources if available) or if there is a limitation to the available resources.
	Predictability of runtime	The system should forecast the execution time for a specific algorithm and a given dataset.
Transparency	Explanation	The complexity of blind classification prevents final users to understand the limitations and boundaries of the models for particular use cases. The definition of rules and explicit models that could help understanding the results could help. Users define such rules or explicit models and the system executes them coordinately measuring differences.
	Provenance	To annotate algorithms and results to trace the processing that has been applied to produce any piece of data, which is extremely important for the revocation of permissions and the fulfilment of the GDPR.
	Accountability	To record a historic log of actions, although it does not need to be accessible by the user.

Table 2 – Trustworthiness Properties

4. USE CASE PRELIMINARY DESIGN

Applications include the necessary components to provide execution and storage. Applications are not only final applications but a bundle that consists of the application logic, the job manager and the storage manager. These components are described as code in a descriptive file and deployed on the cloud resources. The orchestrator service will take the application descriptions and deploy them on the federated infrastructure, by composing compute, storage and network resources. The job manager will be in charge of submitting jobs and managing the resources required in the federation. The job manager and the Storage manager must be black boxes used by the application developers to compose their applications, exposing just the endpoints to access the resources. These services must fulfil all the requirements of elasticity and deployment. The deployment of processing resources can be performed across the federation on different sites.

At design time, ATMOSPHERE will provide a mean to register the application on a system that will perform the analysis of the metrics that apply at design time. The application developer will provide some of these metrics, and some other metrics are generic and available on the platform. The platform also analyses commodity job managers and Storage managers. A score per trustworthy property is obtained, which qualifies the application for production.

At runtime, ATMOSPHERE analyses the dynamic trustworthy features, triggering the actuators associated with specific conditions. The monitoring system gathers the measures from the probes installed on the resources. The monitoring process must be automatic.

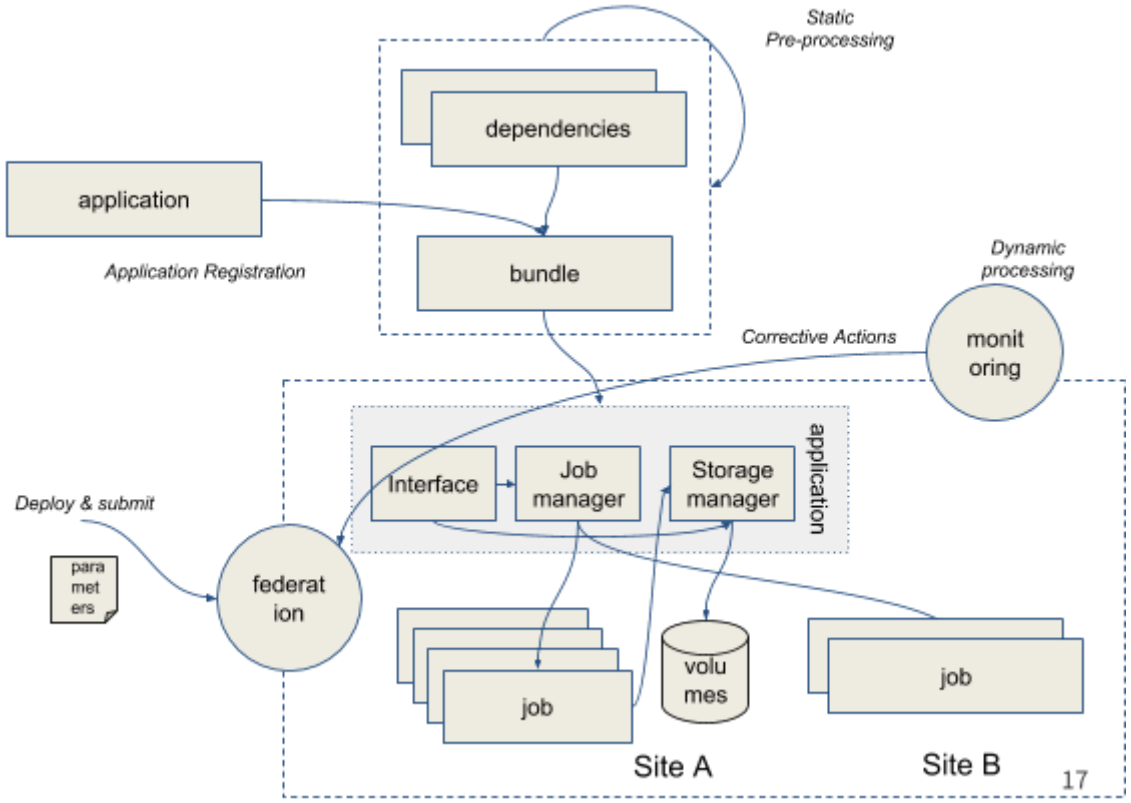

Figure 1 – Application architecture

5. USER STORIES

This section describes the user stories as the actions that application developers and application managers expect to perform on ATMOSPHERE. The user stories related to application developers and application managers are also included in deliverable D4.1, and this deliverable extends this description with the user stories for the final users (section 5.1) and the user stories related to the trustworthy features, which are described in section 5.2.

5.1. Basic user stories

This section describes the user stories for application managers, application developers and final users, from a high-level point of view.

First, we describe the use cases for application managers. Application managers require federation credentials and access to application releases, either as deployment documents or as container images. In this scenario, an application manager must be able to:

- US1.1: Require the deployment of an application defined as an application description (Mandatory).
  + Input: Federation credentials, application description, application requirements.
  + Use case: By using the federation credentials the application manager uses the TDPS to deploy and configure the application provided by the application description. The application requirements guide the cloud federation system to identify the rightmost cloud resource (alternatively, this information can be directly provided by the application manager). The cloud infrastructure will ensure that specific resource configurations are made available in the resources provided. In particular, if the application manager requests (directly or indirectly) GPGPUs or SGX capabilities, those should be available in the resource. The application may also require the creation of a federated virtual network and plug on and off resources to this network. The application manager should have the right to browse the sites for the available VM images.
  + Output: An infrastructure id to be used as a reference in further calls.
- US1.2: List the infrastructures available for a user (Recommendable).
  + Input: Federation credentials and optionally an infrastructure Id.
  + Use case: The application manager uses the TDPS to query about the status of his/her infrastructures or a given infrastructure. The TDPS provides consolidated information about all the resources allocated to the infrastructure.
  + Comment: The TDPs must deal with the granularity (e.g. Infrastructure Manager can do that).
  + Output: Infrastructure description (resources, sites, tags, resource flavours and configuration info).
- US1.3: Undeploy an infrastructure (Mandatory).
  + Input: Federation credentials and the infrastructure Id to be undeployed.
  + Use case: The application manager uses the TDPS to undeploy all the resources of a given infrastructure.
  + Comment: The TDPs must deal with the granularity (again, Infrastructure Manager can do that).
  + Output: Status of the operation.
- US1.4: Reconfigure an infrastructure (Recommendable).
  + Input: Federation credentials, the infrastructure Id to be reconfigured and new application descriptions.
  + Use case: The application manager uses the TDPS to reconfigure an infrastructure that requires changes (e.g. updating software versions, applying patches, adding new software), without undeploying the services and keeping

the content on the virtual disks that is not subject to the reconfiguration. An actuator plugin can automatically trigger this use case too.
    + Output: Status of the operation.

Secondly, we describe the use cases for application developers. Application developers will be in charge of packaging the applications and defining the trustworthiness requirements.

In this scenario, an application manager must be able to:

- US2.1: Define applications and their dependencies for their deployment in the cloud services (Mandatory). Despite that this will mainly depend on the TDPS, there should be a precise translation of the application requirements into the actual services and resources.
    + Input: Application descriptions as TOSCA/RADL documents.
    + Use Case: The TEP will provide the means (repositories, automatic builds, container templates and deployment rules) to build Docker or LXC container images, or to install and configure such applications on the fly. These TEP services will surely be provided by third parties.
    + Output: Container image or configured VM.
- US2.2: Register a resource image for deployment on a specific site (Mandatory).
    + Input: A reference to an LxC or VM image and federation credentials.
    + Use Case: By using the federation credentials, the application developer registers in the local repository of a specific site the LxC or VM image.
    + Output: A reference to the VM or LxC image.
- US2.3: Provide rules and metrics for the evaluation of trustworthy properties that rely on the application (Mandatory).
    + Input: A set of values that define a measure.
    + Use Case: Some metrics may come directly from the applications (e.g. the application progress). The application developer must have the capability of calling a specific API to register a measure from a metric, which will be stored along with information from the instance, timestamp and context.
    + Output: a measure stored in the monitoring database.
- US2.4: Be able to provide actuator plugins for dynamically changing application's behaviour (Mandatory).
    + Input: A logical expression and a hook to a call in the application code.
    + Use case: The application developer can trigger an actuation within the application according to external events (e.g. an application may decide to increase the timestep or the rendering quality to ensure meeting a deadline). The application developer registers a function call to be triggered by the monitoring system when a specific condition (provided by the application developer) is met.
    + Output: The execution of the call.

Finally, we describe the user stories expected by a final user. It is important to outline that although application developers and application managers are generic profiles, final users are closely related to the medical use case. Final users only access ATMOSPHERE through the applications.

A final user should be able to:

- US3.1: Upload / browse / remove / download data on the platform (Mandatory).
  + Input: Data in different formats.
  + Use case: A user should upload data using standard protocols and tools. Datastores can be accessed through third-party tools, such as mounting filesystems or accessing through remote database clients.
  + Output: the result of the processing of the data.
- US3.2: Annotate privacy requirements for data (Mandatory).
  + Input: Privacy requirements defined conveniently.
  + Use case: A user specifies the privacy level for the whole dataset or selected parts of it. The definition of privacy requirements is twofold. First, for the potential annotation of produced results, so the privacy restrictions are kept, and second to guarantee that applications that access it has a desired privacy management guarantee.
  + Output: Not applicable.
- US3.3: Conveniently build data analytic jobs (Recommendable).
  + + Input: a data analytics workflow.
  + + Use Case: An analytic data workflow could be as simple as a single processing job or as complex as a Direct Acyclic Dataflow of I/O, filtering, classification or annotation components that may run in parallel or leveraging processing accelerators such as GPGPUs. A user interface must be available for users to code, run and share them efficiently. The platform should provide a way to publish trained classification components as a service.
  + + Output: A ready-to-run workflow.
- US3.4: Run batch applications under given conditions of Quality of Service (Mandatory).
  + Input: Data and a registered data analytic program.
  + Use Case: Applications are defined and implemented y Application developers, so final users are merely users of such applications. The end-user can build up workflows by linking different steps as described in the previous use case. The end-user can run such jobs in batch mode, and results could be retrieved later on. The end-user can specify Quality of Service constraints (see next section) for the execution.
  + Output: A job identifier to track its progress.
- US3.5: Monitoring the batch jobs submitted (Mandatory).

    + Input: A job identifier.
    + Use Case: By using the job identifier most conveniently, the user can retrieve its progress and eventually the results produced by such job.
    + Output: Status of the job and a handle to retrieve the results if the job has finished.

- US3.6: Run interactive jobs (Recommendable).
    + Input: Data and a registered data analytic program.
    + Use Case: Similarly to a batch job, but in this case, the application executes the job in the foreground assessing the progress interactively.
    + Output: Visualization of the progress of the job.

- US3.7: Obtain a trustworthiness score (Mandatory).
    + Input: Data or component asset.
    + Use Case: Retrieve the scores of the different trustworthiness properties.
    + Output: The trustworthiness score.

5.2. User stories with respect to the trustworthiness properties

This section describes the user stories related to the trustworthiness features. From the use case, we understand that the platform will support not all the properties, but this deliverable presents a more general study to define a more generic design that could be extended for further use cases.

5.3. Security and privacy use stories

The applications developed in the platform must be secure, ensuring that access to sensitive data is properly protected. Along with the functionality of the security infrastructure, we identify two security requirements and four privacy requirements related to the trustworthiness.

- US4.1: To perform an a priori evaluation of the vulnerability (Mandatory).
    + Input: the source code (preferred) or the binaries of the application and some metadata to characterise the application.
    + Use Case: At design time, the system verifies that the application does not miss a set of known vulnerability tests.
    + Output: The platform will provide a vulnerability score to the application developer.

- US4.2: To ensure high test coverage (Recommendable).
    + Input: Unit tests for the components of the application.

  - Use Case: The user must provide a set of unit tests ensuring that the application is appropriately validated.
  - Output: The metric, in this case, will be the percentage of code that the unit tests run through.

- US4.3: The requirement for a protected execution environment (Mandatory).
  - Input: An option indicating this requirement.
  - Use Case: If this is required, the job will run inside an SGX enclave automatically encrypting all the data in memory. If any resource available has this capability, the job must fail. This requirement may come when the application or component is accessing personal data to avoid the legal obligation of informing users in case of security threats, as well as to reduce privacy exposure risk.
  - Output: Not Applicable.

- US4.4: Privacy annotation propagation (Mandatory).
  - Input: Privacy risk score
  - Use Case: The system transfers the privacy risk score of the application to the outcomes of the processing, potentially increasing this risk depending on the score of the rest of the source data and the risk score of the application. For example, an application that extracts unique features from images and relates them to socioeconomic and demographic factors can produce data that could lead to re-identification. This property also requires data to be annotated concerning privacy.
  - Output: Final privacy risk score.

- US4.5: Automatic privacy annotation for classification model (Recommendable).
  - Input: A dataset and a classification model.
  - Use Case: The classification models should check the relevance of individual subjects in the model. The platform should provide the means to analyse how the classification model varies when removing individual cases or critical fields from training.
  - Output: A score based on the relevance of sensitive data in the model.

- US4.6: Dynamic detection of changes on the behaviour of authorised users (Recommendable).
  - Input: Not applicable.
  - Use Case: Outlier detection of behaviours that could lead to identifying risky patterns even if no security break happens. This approach could minimise risks of privacy leaks by early identifying security threats and automatically executing corrective measures.
  - Output: Automatic actions restricting the permissions to specific users or data stores.

- US4.7: Isolation of different jobs (Mandatory).

+ Input: Non-applicable.
+ Use Case: The performance to be equivalent to an isolated and competitive infrastructure (shared with other processes).
+ Output: Non-applicable

5.4. Stability and Performance use stories

The applications developed in the platform must be stable and scalable. Stability is a property that relates to degradation of performance, the reproducibility of the results under different conditions, and the capability of forecasting some of metrics of the application, such as execution time or resource consumption.

- US4.8: No degradation of performance over time (Recommendable).
  + Input: Significant test cases plus a threshold per resource type.
  + Use Case: The application should not degrade memory or processing performance over time above a specified threshold. Periodically, the platform must run the test cases to evaluate the degradation of performance.
  + Output: historical information about the resource usage and the execution time of the tests.
- US4.9: Reproducible deployment (Recommendable).
  + Input: Significant test cases and a model
  + Use Case: The use case application trustworthiness (performance and other trustworthiness scores) to be predictable when deployed in different infrastructures.
  + Output: The expected execution time for a given data and allocation of resources.
- US4.10: Stable sensitivity and specificity (Recommendable).
  + Input: Significant test cases.
  + Use Case: The use case application accuracy should not vary in different deployments (containers, VMs, a different number of resources, GPGPUs).
  + Output: The relative difference among different platforms.
- US4.11: Minimisation of runtime (Mandatory).
  + Input: An application model.
  + Use Case: If the application supports it, ATMOSPHERE should provide means to minimise execution runtime by using distributed resources or specific devices such as GPGPU. The procedure should be convenient for the user. For example, the user could select if runtime should me minimised (using his/her complete quota of resources and specific resources if available) or if there is a limitation to the available resources.
  + Output: The execution of the application under the given conditions.
- US4.12: Sustained Quality of Service (Mandatory).

+ Input: The expected QoS
+ Use Case: The performance to be equivalent to an isolated and competitive infrastructure (shared with other processes).
+ Output: Real QoS.

5.5. Fairness and Transparency use stories

The most innovative metrics are the ones related to the fairness and transparency of the models. The interpretation of the machine learning models is sometimes tricky and ethically unacceptable results may become unnoticed.

Fairness relates to understanding the degree of neutrality of the models for producing results that are not discriminating subjects. Transparency focuses on providing additional information to understand the outcomes obtained.

- US4.13: Non-discrimination based on gender or race (Recommendable).
  + Input: A model, training data and the race and/or sex fields.
  + Use Case: The platform should provide the means to evaluate the potential discrimination of the results based on gender or race (no other personal features are considered according to the nature of the data). This property can be achieved by analysing the results of the model separately for the different subgroups and considering all the subgroups. A significant difference may be reasonable when the problem has a direct correlation with sex or race.
  + Output: A neutrality score.
- US4.14: Explanation of the results (Recommendable).
  + Input: A pair explicit and implicit models.
  + Use Case: The complexity of blind classification prevents final users to understand the limitations and boundaries of the models for particular use cases. The definition of rules and explicit models that could help understanding the results could help. Users define such rules or explicit models and the system executes them coordinately measuring differences.
  + Output: a variance score among models and the consolidated results.
- US4.15: Provenance of the data produced (Mandatory).
  + Input: Unique identifiers for data and algorithms.
  + Use Case: The platform must annotate algorithms and results to trace the processing that has been applied to produce any piece of data, which is extremely important for the revocation of permissions and the fulfilment of the GDPR.
  + Output: Provenance history.
- US4.16: Accountability of the executions (Mandatory).
  + Input: Non-applicable.

+ Use Case: To record a historic log of actions, although it does not need to be accessible by the user.
+ Output: Auditable log information.

6. REQUIREMENTS

The user stories defined above can will be translated into a set of technical requirements that must be fulfilled by the different layers. These requirements must map to the requirements obtained in other layers. As the documents with the requirements from the different layers are due to PM6 (except for the cloud and container management layer, which is already submitted), the analysis will be performed after the release of this deliverable.

This analysis also tries to map requirements with the specific layers, describing the expectations for each layer. This analysis should not be considered as an implementation requirement, but as additional information to describe the requirements.

	Trustworthiness Evaluation Platform	Cloud & Container Services Management Layer	Distributed Trustworthy Data Management Services	Trustworthy Data Processing Services	Application layer
US1.1: Deployment of applications described as code	Matchmaking of trustworthiness properties to resources.	Coherent and Transparent management of resources across the federation. The capability of using specialised resources. Translation of trustworthy properties into deployment information.	Potentially involving the deployment of a storage system.	A DevOps service capable to translate application descriptions into deployment actions.	Describe the application and the dependencies accordingly.
US1.2: List the infrastructures available for a user.		Coherent and Transparent management of resources across the federation.		Consolidated view of the resources deployed.	
US1.3: Undeploy an infrastructure.		Coherent and Transparent management of resources		Consolidated management of individual resources	

		across the federation.		belonging to an application.	
US1.4: Reconfigure an infrastructure.	As US1.1.	As US1.1.	Storage system description should be compatible with reconfiguration	Global reconfiguration of the application deployed.	Application description should be compatible with reconfiguration.

Table 3: Analysis of Application Manager User Stories per Layer

	Trustworthiness Evaluation Platform	Cloud & Container Services Management Layer	Distributed Trustworthy Data Management Services	Trustworthy Data Processing Services	Application layer
US2.1: Define applications and their dependencies for their deployment in the cloud services.	Catalogue of existing certified components and their trustworthiness score.	Translation of trustworthy properties into deployment information.	Potentially involving the deployment of a storage system.	A DevOps service that can translate application descriptions into deployment actions.	Describe the application and the dependencies accordingly.
US2.2: Register a resource image for deployment on a specific site.	Trustworthiness scores.	The capability of remotely accessing the resource image store (e.g. glance or Docker repository)		Automatic building of images.	
US2.3: Provide rules and metrics for the evaluation of trustworthy properties that rely on the application.	Define API for registering metrics. Provide a monitoring database.	Deal with the generic metrics that relate resources managed by the cloud infrastructure.	Deal with the generic metrics that relate resources managed by the cloud storage.		Provide with probes and rules.
US2.4: Be able to provide actuator plugins for dynamically changing application's	Define API for registering actuators and rules. Provide a monitoring	Generic actuators related to the cloud services.	Generic actuators related to the storage.	Generic actuators that rely on changes related to the deployment.	Application-specific actuators.

behaviour.	actuator.				

Table 4: Analysis of Application Developer User Stories per Layer

	Trustworthiness Evaluation Platform	Cloud & Container Services Management Layer	Distributed Trustworthy Data Management Services	Trustworthy Data Processing Services	Application layer
US3.1: Upload / browse / remove / download data on the platform.			Provide best practices.	GUI if needed.	
US3.2: Annotate privacy requirements for data.	Annotation of privacy metadata		Annotation of privacy metadata	GUI for privacy requirements.	
US3.3: Conveniently build data analytic jobs.	Annotate trustworthiness properties.			GUI for analytic jobs.	
US3.4: Run batch applications under given conditions of Quality of Service.	Annotation of QoS Metadata. Metric collection.	Provide the necessary resources.		Provide a job scheduler able to deal with the infrastructure.	
US3.5: Monitoring the batch jobs submitted	Metric collection.			As US3.4	
US3.6: Run interactive jobs	Metric collection	As US3.4 for this type of jobs.		As US3.4 for this type of jobs.	
US3.7: Obtain a trustworthiness score	Evaluation of design time metrics. Collection of metrics.	Evaluation of dynamic metrics regarding cloud services.	Evaluation of dynamic metrics regarding storage.	Evaluation of dynamic metrics regarding processing	Evaluation of dynamic metrics regarding applications.

				services.	

Table 5: Analysis of Final User Stories per Layer

	Trustworthiness Evaluation Platform	Cloud & Container Services Management Layer	Distributed Trustworthy Data Management Services	Trustworthy Data Processing Services	Application layer
US4.1: To perform an a priori evaluation of the vulnerability.	Evaluation of design time metrics.	Provide resources if necessary.			Provide application code / binaries.
US4.2: To ensure high test coverage.	Run test cases and provide a score.	Provide resources if necessary.			Provide test cases.
US4.3: The requirement for a protected execution environment.	Annotate it in the metadata.	Provide SGX resources through VM.			
US4.4: Privacy annotation propagation.	Annotate it in the metadata of the products.		Annotate it in the metadata of the products.	Annotate it in the metadata of the products.	
US4.5: Automatic privacy annotation for classification models.		Manage metrics.		Trigger the computation of the metric.	Describe privacy and procedures.
US4.6: Dynamic detection of changes in the behaviour of authorised users.	Manage Metrics and actuators.		Outlier detection.		Describe reasonable risk thresholds.
US4.7: Isolation of different jobs.	Evaluate and register isolation.	Provide isolation at container and VM level.			Describe reasonable risk.thresholds.

US4.8: No degradation of performance over time.	Evaluate and register performance.				Provide tests.
US4.9: Reproducible deployment.	Evaluate at design time.			Platform agnostic application description and deployment service.	Provide tests.
US4.10: Stable sensitivity and specificity.	Evaluate dynamically.				Provide tests.
US4.11: Minimisation of runtime.	Collect metadata.	Application modelling and Matchmaking of configuration parameters.			Provide target.
US4.12: Sustained Quality of Service	Evaluate dynamically, trigger actuators.	Actuators at Cloud services layer.	Actuators at Storage services layer.	Actuators at Processing services layer.	Actuators at Application services layer.
US4.13: Non-discrimination based on gender or race.	Manage metrics.			Trigger the computation of the metric.	Describe fairness and goals.
US4.14: Explanation of the results.	Manage metrics.				Provide descriptive models.
US4.15: Provenance of the data produced.	Collect metadata from applications and data.		Annotate processing actions on data.	Annotate processing actions on data.	
US4.16: Accountability of the executions.	Collect metrics.	Log actions at cloud service level.	Log actions at the storage level.	Log actions at cloud processing level.	

Table 6: Analysis of User Stories for trustworthiness properties per Layer

The analysis of the previous User Stories ended up with the following list of requirements. Those requirements are associated with the different User Stories. Previous sections introduced additional details for those User Stories and therefore for the requirements. The list of requirements has been consolidated to facilitate planning and interoperability.

- RP1. Application Delivery (US1.1, US2.1, US2.2, US4.13, US4.14). Applications should be described as code, including the software dependencies, the configuration actions

and the deployment-time customizable parameters. Reusability of components should be maximised. The description of the applications should include if required, the means to compute the application-specific trustworthiness properties and the adaptation mechanisms for such properties.

- RP2. Design-time application analysis (US2.1, US2.3, US4.1, US4.2, US4.13, US4.14). The ATMOSPHERE platform analyses a priori the applications by executing vulnerability tests, unit test coverage, deployment tests and other tests provided by the components used in the application or the application itself. Measures such as fairness, explanation or automatic privacy annotation, which involve executing different training approaches require the application developer to annotate the sensitive data fields.
- RP3. To deploy a single-resource application across the federation (US1.1, US4.3, US4.7, US4.9). The user may include restrictions on the site to deploy an application expressed as defined in RP1 and validated as described in RP2. The deployment may require specific resources (e.g. the execution of the application or parts of it within protected environments, the use of accelerators, etc.). The principles of isolation and reproducibility must be considered.
- RP4. Deployment of an application involving a job manager across the federation (US1.1, US3.6, US4.3, US4.7, US4.9). The user may include restrictions on the site to deploy an application expressed as defined in RP1 and validated as described in RP2. The deployment may require specific resources (e.g. the execution of the application or parts of it within protected environments, the use of accelerators, etc.). The principles of isolation and reproducibility must be considered.
- RP5. QoS Guarantees (US1.4, US2.3, US2.4, US3.4, US4.8). ATMOSPHERE treats especially QoS guarantees, understood as the expected performance of a job. ATMOSPHERE will allocate the necessary resources for executing a job with performance requirements (such as meeting a deadline or ensuring the allocation of a given amount of resources to a job). Potentially, the QoS can depend on the application (e.g. when the progress of the application is explicitly available). Actuators can act as infrastructure level to reconfigure the infrastructure to provide with the necessary resources. The actuators must be integrated with the job management systems to ensure the fulfilment of the QoS for individual jobs. Performance must be stable along time.
- RP6. Trustworthiness properties annotation (US3.2, US4.3, US4.4, US4.5, US4.13, US4.14, US4.15). ATMOSPHERE focuses on measuring and improving the trustworthiness of applications and services. The user will provide the annotations of the trustworthiness metrics to evaluate them. The user will identify sensitive fields, will define privacy restrictions if known a priori, QoS goals, fairness targets and explicit models for the explanation. This information will be propagated (and potentially amended) to the resulting data by the processing layers.

- RP7. Dynamic adaptation (US4.5, US4.6, US4.8, US4.11, US4.12). Automatic adjustment relies on the adaptation mechanisms. ATMOSPHERE defines adaptation mechanisms at the level of the different layers, and additional adaptation mechanisms are defined at the application level. Adaptation refers primarily to privacy (execution on protected environments, site restriction or execution abortion due to risks exceeding a threshold) and performance (sustained QoS).
- RP8. Trustworthiness monitoring (US3.5, US3.7, US4.8, US4.10, US4.11, US4.15, US4.16). ATMOSPHERE will provide trustworthiness evaluation both at design time and runtime. The monitoring system will evaluate the measures and collect the metrics for the different trustworthiness properties, including all layers and applications. The metrics will relate to privacy risks, performance, resource allocation, model accuracy, accountability and provenance.
- RP9. Resource Listing (US1.2). ATMOSPHERE will provide means for listing the resources deployed for a specific application or user.
- RP10. Resource Undeployment (US1.3). ATMOSPHERE will provide means to undeploy the resources deployed for a specific application or user, in a convenient (e.g. all resources of an application, all resources of a user, specific resources) and secure way.
- RP11. To provide persistent POSIX-like accessible storage (US1.1, US3.1). The applications and jobs expect to work with POSIX filesystems. ATMOSPHERE should provide applications with this type of storage. The volumes should be persistent and reusable in different deployments.
- RP12. To provide temporary POSIX-like accessible storage (US1.1, US3.1). Similarly to RP11, but considering volatile storage for temporary data.
- RP13. To provide relational database instances (US1.1, US3.1). ATMOSPHERE will offer components to the applications to instantiate relational database instances, correctly configured with the security and network parameters of the whole application.
- RP14. To provide NoSQL database instances (US1.1, US3.1). This requirement is equivalent to RP13 but related to NoSQL databases.

7. CONCLUSIONS

This document has described the pilot use case that will be used as a demonstrator of the ATMOSPHERE platform. This use case focused on the remote processing of sensitive medical information data, stands out as a representative use case of an application requiring a high degree of trustworthiness. The procedure for analysing requirements has involved all representatives of the technical workpackages in retrieving the information about the expectancies of the ATMOSPHERE platform. The application developers drove the requirement analysis.

The use case focuses on Rheumatic Heart Disease, a heart disease caused by poorly treating a set of infectious diseases that cause damages in the heart walls and valves. If detected, the infections can be easily managed, but if it remains undetected, it can cause severe morbidity. The project will use echocardio images obtained from children in rural areas to automatically evaluate the risk of RHD according to the World Heart Organization parameters.

The project identifies three main actors for the pilot case: Application developers, application managers and end-users. The document defines 31 needs and user stories for the three profiles, related to the management of resources, storage, computing, security and other trustworthiness properties. This analysis led to identifying 14 requirements that will be used as the basis for the development of the different ATMOSPHERE layers.

8. ACRONYMS AND ABBREVIATIONS

Acronym	Definition
AOB	Any Other Business
ATMOSPHERE	Adaptive, Trustworthy, Manageable, Orchestrated, Secure, Privacy-assuring, Hybrid Ecosystem for REsilient Cloud Computing
CA	Consortium Agreement
CNPq	National Council for Scientific and Technological Development of Brazil
CooA	Coordination Agreement
CTIC	Centro de pesquisa e desenlvolvimento em tecnologias digitais para informação e comunicação (Center for Research and Development in Digital Technologies for Information and Communication)
DoA	Description of Action
EC	European Commission
EEC	External Expert Committee
EU	European Union
GA	Grant Agreement
ICT	Information and communications technology
INCT	Instituto Nacional de Ciência e Tecnologia (National Institute on Science and Technology)
IT	Information Technology
KPI(s)	Key Performance Indicator(s)
M	Month
MCT	Ministry of Science and Technology
PC	Project Coordinator
PEB	Project executive Board
PM	Person Month
PMB	Project Management Board
RNP	Rede Nacional de Ensino e Pesquisa
RTB	RTD Board
RTD	Research Technology Development
WP	Work Package
WPL	Work Package Leader

Table 7 – Acronyms and Abbreviations