Project Acronym: STORM CLOUDS

Grant Agreement number: 621089

Project Title: STORM CLOUDS – Surfing Towards the Opportunity of Real Migration to CLOUD-based public Services

Deliverable D2.1
Storm Clouds Platform Requirements and Specification

Work Package: WP2

Version: 2.0

Date: 23/12/2015

Status: Project Coordinator Accepted

Dissemination Level: PUBLIC

Authoring

Role	Name	Organisation
Edited by	Marco Consonni	Hewlett Packard Italiana
Author	Marco Consonni	Hewlett Packard Italiana
Author	Pasquale Panuccio	Hewlett Packard Italiana
Author	Vincenzo Cultrera	Hewlett Packard Italiana
Reviewed by	Panagiotis Tsarchopoulos	Aristotelio Panepistimio Thessalonikis
Reviewed by	Julian Arroyo Alvarez	Ayuntamiento de Valladolid
Reviewed by	Agustin Gonzales-Quel	Ariadna

Version Control

Modified by	Date	Version	Comments
M. Consonni	03-03-2014	DRAFT	ToC Insertion
M. Consonni	07-04-2014	0.1	Ready for the 1st Internal Review
M. Consonni	10-04-2014	0.2	Executive Summary and Summary and Conclusions, added
M. Consonni	30-04-2014	1.0	Version 1.0
M. Consonni, S. De Domenico	16-12-2015	1.1	Recovery actions after EUC review: - SCP and SCPP abbreviations added - Non-functional requirements (aka. "sizing information") for services coming from URENIO added - Description of Have Your Say (a.k.a. Participação Pública PPGIS - Public Participation), service of the municipality of Agueda, added
M. Consonni	23-12-2015	2.0	Version 2.0

Project Presentation

Surfing Towards the Opportunity of Real Migration to Cloud-based public Services (STORM CLOUDS) [1] is a project partially funded by the European Commission within the 7th Framework Program in the context of the Capital Improvement Plan (CIP) project (Grant Agreement No. 621089) [2].

The project has the objective of exploring the shift to a cloud-based paradigm for deploying services that Public Authorities (PAs) currently provide using more traditional Information Technology (IT) deployment models. In this context, the term "services" refers to applications, usually made available through Internet, that citizens and/or public servants use for accomplishing some valuable task.

The project aims to define useful guidelines on how to implement the process of moving application to cloud computing and is based on direct experimentation with pilot projects conducted in, at least, the cities participating to the consortium.

STORM CLOUDS will also deliver a consolidated a portfolio of cloud-based services validated by citizens and Public Authorities in different cities and, at the same time, general and interoperable enough to be transferred and deployed in other European cities not taking part in the project. This portfolio will be mainly created from applications and technologies delivered by other CIP Policy Support Program (CIP-PSP) and Framework Program 7 (FP7) projects, as well as resulting from innovation efforts from Small and Medium Enterprises (SMEs).

The project is lead by the following consortium:

Member	Role/Responsibilities	Short Name	Country
Ariadna Servicios Informáticos, S.L.	Co-ordinator	ASI	Spain
Hewlett Packard Italiana S.r.l.	Participant	HP	Italy
EUROPEAN DYNAMICS Advanced Systems of Telecommunications, Informatics and Telematics	Participant	ED	Greece
Research, Technology Development and Innovation, S.L	Participant	RTDI	Spain
Aristotelio Panepistimio Thessaloniki	Participant	AUTH	Greece
Alfamicro Sistemas de Computadores LDA	Participant	Alfamicro	Portugal
Manchester City Council	Participant	Manchester	United Kingdom
Ayuntamiento de Valladolid	Participant	Valladolid	Spain
City of Thessaloniki	Participant	Thessaloniki	Greece
Câmara Municipal de Águeda	Participant	Águeda	Portugal

For more information on the scope and objectives of the project, please refer to the Description of Work (DOW) of the project [3].

Executive Summary

Work Package 2 (WP2) of the Storm Clouds project is aimed at designing and implementing the reference architecture for the Storm Clouds Platform (SCP), the cloud platform infrastructure for hosting application services selected for being ported to cloud. SCP supplies computational resources that are allocated/de-allocated on-demand following a "as-a-Service" cloud computing paradigm.

This document defines the requirements and specification for SCP and it is organized into the major sections described below.

Section 1 describes the **business context** where the platform is used highlighting the business entities (A.K.A. business actors) and their main roles and responsibilities. **Section 2** is lists the **applications** candidate to be ported to cloud during the project, providing some basic functional and technical information with the objective of deriving requirements for the Storm Clouds Platform. There are several reasons for assessing the candidate applications at this stage both technical and business-wise. For example, it is worth understanding the kind of functions applications implement in order to find out if and how they can be reused in other contexts. As a matter of fact, one of the objectives of the project is to "*deliver a consolidated cloud-based service portfolio [...omitted...] general and interoperable enough to be transferred and deployed in other European cities*" [3]. From the technical view point, the assessment provides important insights on the nature of the proposed application services like the type of workload (e.g. on-line, batch, web-based, mobile, etc.), indications on security issues, an estimation of the amount of resources required to run, etc. This information is mainly gathered for designing a cloud platform that better fit the application requirements. **Section 3** defines the **Storm Clouds Platform requirements and specification**. It starts with a discussion of the National Institute of Standard Technology (NIST) definition of cloud computing and tries to find what aspects are to be addressed in the project. The reason for this exercise is that NIST provides a good baseline for discussing about what cloud computing is and how to best use it, without proposing a "pre-canned" product-oriented solution that is not in the objective of the project. Therefore, it's worth "challenging" the NIST definitions with the objective of eliciting requirements applicable to the project independently on the actual technical solutions that will be implemented. The results of the exercise are arranged in the cloud platform functional decomposition of the cloud platform (a logical model showing how the functions relate one another) and a list of requirements. **Section 5, Storm Project Methodological Framework,** briefly reports some relevant information about the Storm Clouds project framework in order to highlight aspects that have direct impact on the design of the SCP. Section *5* summarizes the document topics and highlights the most relevant aspects for the upcoming project activities.

Table of Contents

List of Figures

List of Tables

Abbreviations

Acronym	Description
API	Application Programming Interface
CLI	Command Line Interface
DBMS	DataBase Management System
DNS	Domain Name System
DRDB	Distributed Replicated Block Device
ERP	Enterprise Resource Planning
GIS	Geographic Information System
GRE	Generic Routing Encapsulation
GTFS	General Transit Feed Specification
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
KML	Keyhole Mark-up Language
KVM	Kernel-based Virtual Machine
IaaS	Infrastructure as a Service
IT	Information Technology
LXC	LinuX Containers
NIST	National Institute of Standards and Technology
N/A	Not Available
PC	Personal Computer
PaaS	Platform as a Service
REST	REpresentational State Transfer
RSS	Really Simple Syndication
SaaS	Software as a Service
SCP	Storm Clouds Platform
SCPP	Storm Clouds Project Platform
SSL	Secure Socket Layer
TCP-IP	Transmission Control Protocol – Internet Protocol (suite)
URL	Uniform Resource Locator
VM	Virtual Machine
VPN	Virtual Private Network
WLAN	Wireless Local Area Network

1 Business Context

The following diagram depicts the context where the Storm Clouds Platform is used and shows the entities that have some sort of interaction with it:

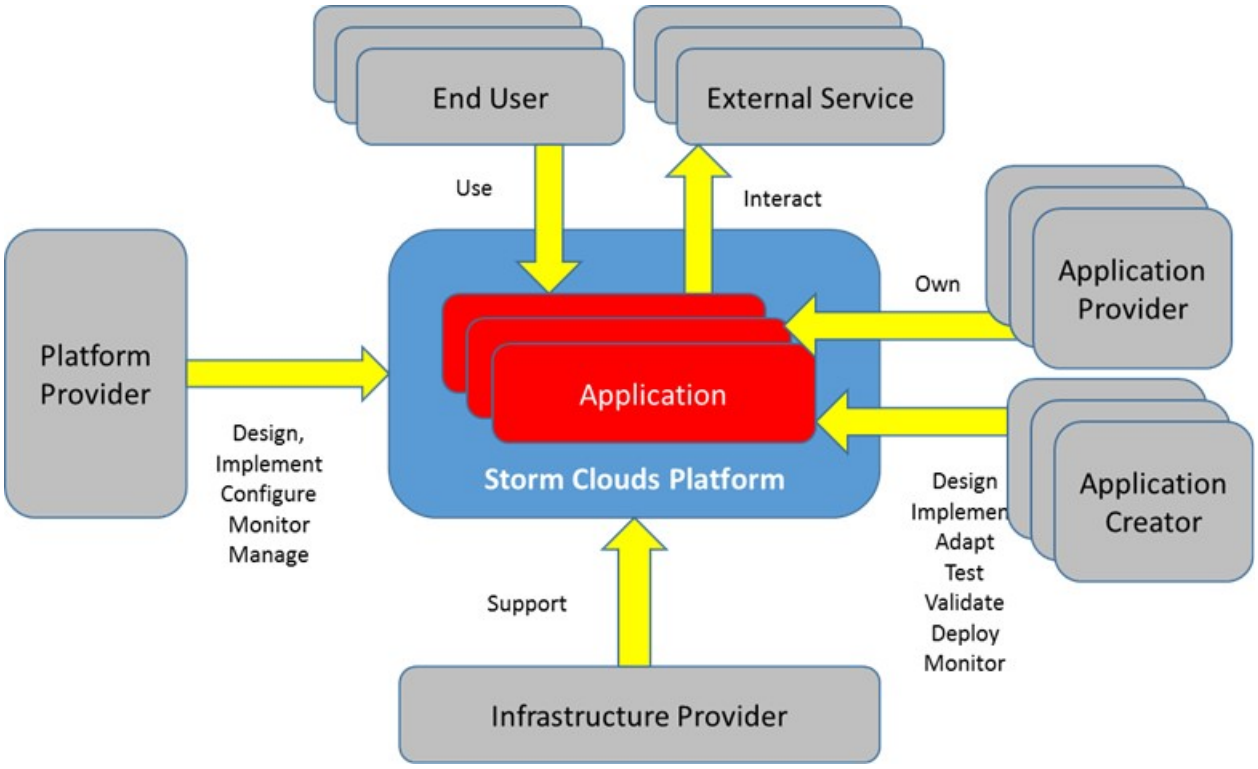

Figure 1-1 – Business Context

1.1 Storm Cloud Platform

Storm Clouds Platform (SCP) is the "digital space" where Storm Clouds services are hosted; it is a cloud computing platform providing resources made available for running applications.

In the project context, an actual SCP environment is implemented for being shared by the participants of the consortium; it is built on top of a physical computing infrastructure and provisioned to partners via Internet. This actual SCP implementation is also called Storm Clouds Project Platform (SCPP) for distinguishing it from SCP that is intended to refer to the cloud platform reference architecture that could be deployed in other contexts.

In fact, it's possible to implement other SCP deployments when some project participants (e.g municipalities) require that applications run in a different, maybe "more controlled", environment. This might be preferred and/or strictly required by security and privacy regulations on the information managed by the applications. In such a case, SCP design can be totally or partially reused for the implementation of an SCP instance. It's up to the implementer to provide the physical infrastructure for running SCP as well as to satisfy the governance and the operational activities that the SCP implementation and the applications running on top of it might require.

1.2 Applications

Applications are software artefacts implemented for providing some valuable functions to users like citizens and/or public servants: for example a web based service designed for planning a guided tour in a city or a digital marketplace where small businesses can publish their offers. Applications can interoperate each other: for example the two above mentioned applications might interoperate with a third application for visualizing a local map.

Applications deployed during the project are selected from the set of applications assessed in this document; they usually require modifications in order to run on SCP but the SCP is designed for minimizing interventions and adaptations.

1.3 End Users

End Users are the human beings the applications are implemented for. They can be citizens and/or public servants who, taking advantage of the implemented functions can accomplish some valuable task.

For example, if an application is designed for supporting on-line surveys, end users are the citizens who use the application for expressing their opinions on certain topic. In another example, if the application is designed for managing municipal master plans, end users are public servants who use it for urban planning, inventory, etc; end users are also citizens who are given the possibility to examine public documents. In this perspective, all the consumers of all the applications hosted by SCP are collectively called end users.

SCP in itself does not implement any significant function for end users; actually it provides all the functionalities the end users use through the hosted applications deployed on the platform.

1.4 Application Creators

Application Creators are the entities in charge of implementing the applications hosted in the SCP. From the project perspective, application creators are organizations like public administrations and/or SMEs that implement, adapt and manage the applications.

SCP provides application creators with functions for deploying applications using some computational resources (e.g. virtual machines, virtual disk space, etc.) and for monitoring and controlling their lifecycle in the platform.

1.5 Application Providers

Application Providers are the owners of the applications. In the project context, public administration entities (e.g. municipalities) are the application providers. They are in charge of selecting the applications to run in the SCP and, in some cases, adapting the services to their own requirements.

1.6 Platform Provider

The Platform Provider is the organization in charge of implementing the SCP. In the project context, the platform provider has the responsibility of designing SCP by selecting the technology used for implementing, monitoring and operating the platform.

1.7 Infrastructure Provider

The Infrastructure Provider provides the physical resources for implementing the Storm Cloud Platform. It is in charge of providing computing resources like physical machines, disk storage and network connectivity used for actually deploying the software for implementing SCP. The infrastructure provider is also in charge of providing the physical place where the computing resources are located (i.e. data centre(s)) along with power & cooling facilities, physical security infrastructure (e.g. data centre access control), etc.

In the Storm Clouds project, this role is played by Hewlett Packard for what concerns the implementation of a shared Storm Cloud Platform but it can also be covered by other entities (e.g. public administrations) in case, for some business, technical, governance, operational reasons SCPP might result as an inconvenient choice for hosting some application. In this perspective, the SCP should be "portable" meaning that it should be designed for running on a "conventional data centre".

1.8 External Services

External Services are "digital services", running outside the Storm Cloud Platform, that provide functions used for implementing applications.

For example, an application implementing a virtual tour of a city might use some service, available on Internet through a web service interface (e.g. Google Maps), for creating maps. In this example, Google Maps is an external service because although it is used for implementing the virtual city tour application is not hosted in the Storm Clouds Platform. As another example, an on-line survey application might require that citizens authenticate themselves before signing the petition. The authentication could use a digital service, provided, as a web service, by the local administration that, for security reasons, is deployed on a "safe" digital workplace (e.g. on an on-premise data centre owned and/or controlled by the public authority). In this example, the authentication service is an external service.

The Storm Cloud Platform shall make it possible application services and external service interoperate in order to provide end users with the functions implemented by the applications.

2 Application Service Assessment

This chapter is the result of the assessment of the applications candidate to be ported to cloud during the Storm Clouds project. It gathers functional and technical information with the objective of defining requirements for the Storm Clouds Platform. The analysis also highlights obstacles to the transformation or the porting of applications to cloud due to technical and functional reasons. From the technical side, some applications might be implemented with legacy technologies or might require licenses for using commercial software products. From the functional view point, issues could be related to the data processed by the applications. Sensitive information, like personal information, could raise privacy and security issues. Both these aspects can make porting to cloud impossible or unpractical.

2.1 Applications

This section reports the list of the candidate applications proposed by the project participants.

The information reported here has been gathered by circulating a questionnaire among the participants and, in some cases, complementing the collected data with information found on the application website, when available.

Each application is described by providing the following information:

- **Proponent – Application**: the name of the partner who proposes the application and the name of the application;
- **Functional Description**: a brief description of the implemented functions and the users of the application;
- **Availability**: the current version of the application and a link to a deployment available on the Internet (if any);
- **Technical Information:**
    - The list of the technologies used for implementing the application (e.g. the operating system, programming languages, database engines, etc.);
    - Resource information like amount of required RAM, disk space, number of vCPUs, etc.
    - Deployment information like the number of servers for running the application, high availability solutions, load balancing solutions, etc.

2.1.1 Municipio de Águeda - Emissão de Plantas de Localização

Emissão de Plantas de Localização is a local map management system implementing visualization, design, search, and print of local maps. It is used by citizens, community groups, technicians and public entities in general.

The application is currently released as a prototype (v. 2.0) available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu
Programming Languages	Javascript, PHP, Java
Databases	PostgreSQL, PostGIS
Web/Application Servers	Tomcat, Apache, Geoserver
Frameworks	ExtJS, NodeJS
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: git **Build Management: -**
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	Value
RAM {GB}	12
Disk Storage {GB}	60

vCPUs	8
Network usage {GB}	N/A
Hits/Month	250
Registered Users	691
Maximum On-line Users	N/A
Average On-line Users	N/A

The application requires two servers: one database server (4 vCPU – 4 GB RAM) and one web server (4 vCPU – 8 GB RAM). These servers are shared together with application Plano Director Municipal (see below). Load balancing and high availability are not implemented.

The application manages sensitive information like name, address and VAT number of the users.

2.1.2 Municipio de Águeda - Plano Director Municipal

Plano Director Municipal is an application that manages municipal maps. Users visualize maps, search point of interests and export map data in KML format. KML format is used to display geographic data in Earth browsers such as Google Earth, Google Maps, and Google Maps for mobile. Users are citizens, community groups, technicians and city hall administrators.

The application is currently released as prototype (v. 1.0) available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	Ubuntu
Programming Languages	Javascript, PHP, Java
Databases	PostgreSQL, PostGIS
Web/Application Servers	Apache, Tomcat, Geoserver
Frameworks	ExtJS, NodeJS
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: git **Build Management: -**
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	**Value**
RAM {GB}	12
Disk Storage {GB}	10
vCPUs	8
Network usage {GB}	N/A
Hits/Month	N/A
Registered Users	N/A
Maximum On-line Users	N/A
Average On-line Users	N/A

The application is deployed on the same servers of *Emissão de Plantas de Localização* (see above).

2.1.3 Municipio de Águeda - HotSpot CMA

HotSpot CMA is a project that allows access to information and communication technologies as a form of democratization. The project has implemented hotspots located in the city area and a captive portal through which users (i.e. citizens) access Internet services.

The application candidate to be ported to cloud is the captive portal that implements a web based authentication interface. It is currently in testing phase (v 1.4) and a demo is available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	Ubuntu
Programming Languages	Javascript, PHP

Databases	MySQL
Web/Application Servers	nginx
Frameworks	ExtJS, JQuery
Application Lifecycle Tools	**IDE**: NetBeans, Aptana Studio **Version Control**: svn **Build Management: -**
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	**Value**
RAM {GB}	2
Disk Storage {GB}	20
vCPUs	2
Network usage {GB}	200
Hits/Month	6000
Registered Users	1500
Maximum On-line Users	100
Average On-line Users	20

The application is deployed on a single server; load balancing and high availability features are performed using VMware virtualization technology. The captive portal interacts with a management portal, a FreeRADIUS server all connected to the same MySQL database instance. The system requires a domain name: *agueda.pt*.

It is worth noticing that users implicitly interact with other systems when accessing Internet (e.g. access points, routers, gateways, etc.). During the assessment, some technical information was provided about these equipment but it is not reported here because these components are not meant to be candidate for being ported to cloud.

2.1.4 Municipio de Águeda - Lime Survey

Lime Survey is an online portal where users can publish and collect responses from questionnaires; it can be used by local authorities to collect feedback from citizens about proposals, events, etc. The application let users to invite group of people to participate in research, keep track of what the survey found, and ensure that each person can only enter once. The application also keeps tracks of participants who has not yet responded, and send them reminder email. Administrator can import lists of names and email addresses and generate a unique token number for each participant.

The application is currently in testing (v. 2.05) and a demo is available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	Linux (generic)
Programming Languages	Javascript, PHP, Perl
Databases	PostgreSQL / MS SQL
Web/Application Servers	Apache
Frameworks	-
Application Lifecycle Tools	**IDE**: - **Version Control**: git **Build Management: -**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	**Value**
RAM {GB}	1
Disk Storage {GB}	160
vCPUs	2
Network usage {GB}	N/A
Hits/Month	N/A
Registered Users	N/A

Type	Value
Maximum On-line Users	N/A
Average On-line Users	N/A

The application is deployed on a single server. Load balancing and high availability implementation are take advantage of the VMware virtualization technology solutions (e.g. VMware HA). The application implements a "tokens batch function" that consist of generating a unique token number for each participant subsequently used to manage the life-cycle of the survey.

It supports both PostgreSQL and MS SQL but the current deployment uses the former one.

It manages sensitive information like users' names and address.

2.1.5 Municipio de Águeda - Have Your Say

Have Your Say is a WebGIS[1] application that allows any citizen to express their opinion on a theme under discussion. Each participation is related with some geographic feature (street, path, building, garden, furniture, etc). Users can start a new discussion topic or add comments to existent topics. Users can upload photographs or documents relevant for the discussion. Each time a new topic is started, edited or deleted, the moderators are notified by email. This application has an administrative interface to manage themes under discussion. Privileged users (either from local administration or from community) can add new themes for discussion.

The application is currently released as a prototype available at address http://euparticipo.cm-agueda.pt.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu
Programming Languages	Javascript
Databases	PostgreSQL + PostGIS
Web/Application Servers	Node.js + MapProxy
Frameworks	ExtJS; OpenLayers; GeoExt
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: git **Build Management:** sencha cmd, npm
Open Source Code Repository	GitHub

The following table reports the sizing information:

Type	Value
RAM {GB}	1
Disk Storage {GB}	20
vCPUs	1
Network usage {GB}	N/A
Hits/Month	1500
Registered Users	80
Maximum On-line Users	50 (estimated)
Average On-line Users	1-5

Currently, the application requires a DNS name and the necessary Apache configuration to launch the server. The server is written on node.js. It requires an external Postgresql database and a Redis server for session storage. To support more than 20 simultaneous users, the default Postgresql configuration must be changed to support more connections to the database. To handle more connections, the current deployment uses the pgbouncer application to manage the database connections.

The application manages sensitive information like name and e-mail. Optionally, users can publish then own picture to work as avatar. The participation of each user can be tracked: every discussion, comment,

[1] A WebGIS application is a Geograohic Information System (GIS) that uses web technologies.

photograph or document can be associated with the person. The application also logs the login and logout operations, recording the date and time, and also the IP address used to access the application.

2.1.6 Ariadna Servicios Informáticos - Co-Labora

Co-Labora is a workflow management application that citizens and city hall employees can use for collaborating, for example for reporting and managing urban incidents.

The application is under development.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Linux (generic), Win Server 2008
Programming Languages	PHP
Databases	MySQL
Web/Application Servers	Apache
Frameworks	Drupal
Application Lifecycle Tools	**IDE**: N/A **Version Control**: svn **Build Management: -**
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	Value
RAM {GB}	N/A
Disk Storage {GB}	N/A
vCPUs	N/A
Network usage {GB}	N/A
Hits/Month	N/A
Registered Users	100000
Maximum On-line Users	2500
Average On-line Users	800

The application is deployed on a single server and implements batch functions for generating usage statistics.

Neither high availability nor load balancing solutions are currently implemented.

It manages sensitive data like citizens' location.

2.1.7 Ariadna Servicios Informáticos – SEDOC

SEDOC is an application managing and tracking physical and digital documents. It's available in the form of web user interface, Windows Phone Application and Windows desktop application.

The application is currently under development (v. 0.91).

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu
Programming Languages	PHP, Java, C#
Databases	MySQL, SQLite
Web/Application Servers	Tomcat
Frameworks	Drupal, NodeJS, Puppet, Vagrant
Application Lifecycle Tools	**IDE**: Eclipse, Visual Studio **Version Control**: svn **Build Management:** Maven, Jenkins
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	Value

RAM {GB}	1
Disk Storage {GB}	10
vCPUs	N/A
Network usage {GB}	0.5
Hits/Month	N/A
Registered Users	N/A
Maximum On-line Users	N/A
Average On-line Users	N/A

The application is deployed on a single server and it is integrated with the client ERP via web services.

The application manages confidential information regarding financial-related contracts of the municipalities.

2.1.8 Ariadna Servicios Informáticos – SIMPLEXT

SIMPLEXT is a text simplification tool for Spanish language that simplifies text from RSS sources and also during the web navigation. It is intended for people with intellectual disabilities and for intensive news consumers.

The application is currently released as a prototype available here (subject to a previous access request).

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	Ubuntu
Programming Languages	Java
Databases	N/A
Web/Application Servers	Tomcat, Apache
Frameworks	-
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: svn **Build Management: -**
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	**Value**
RAM {GB}	6.5
Disk Storage {GB}	N/A
vCPUs	N/A
Network usage {GB}	0.5
Hits/Month	N/A
Registered Users	2000
Maximum On-line Users	50
Average On-line Users	20

The application requires a public domain (*www.simplext.net*) and it is integrated with third-party open source components like GATE, Freeling, MATETools, OpenNLP and LexSis.

2.1.9 Manchester City Council - City Navigator

City Navigator is a fully Open Source, mobile HTML5 public transport journey planner and navigation application for on-the-go use. It leverages data from multiple sources including OpenStreetMap, TfGM, Manchester City Council, and CitySDK. The goal is to take unpredictability away from public transportation and make it more accessible. It is used by people using or plannning to use the public transportation of the city of Manchester.

The application is currently released as a prototype.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	N/A

Programming Languages	Python
Databases	N/A
Web/Application Servers	N/A
Frameworks	NodeJS, JQuery Mobile, Leaftlet, Faye
Application Lifecycle Tools	**IDE:** N/A **Version Control:** git **Build Management: -**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	**Value**
RAM {GB}	N/A
Disk Storage {GB}	N/A
vCPUs	N/A
Network usage {GB}	N/A
Hits/Month	N/A
Registered Users	N/A
Maximum On-line Users	N/A
Average On-line Users	N/A

The web application requires a public domain. Programming languages include Python for backend/server code and CoffeeScript (compiled into JavaScript) for backend and front-end code.

At the time being, the application doesn't manage sensitive information but in future it could be personalised with social media data.

2.1.10 Ayuntamiento de Valladolid - Blue Parking Valladolid

Blue Parking Valladolid is an application that implements an intelligent parking system for the city of Valladolid. The application doesn't require sensors or other infrastructure because information on the available parking spaces is implicitly provided by the users via their mobile devices. Users find a car park and pay directly from smart-phone while inspectors use a specific version of the application to validate the park.

The application is delivered (v. 1.0) and available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	CentOS, Windows Server 2008 Sp2
Programming Languages	Javascript, Java, Python, PL/SQL
Databases	MongoDB, Oracle SEO
Web/Application Servers	Tomcat
Frameworks	NodeJS, socket.io, mongojs, redis, poolee
Application Lifecycle Tools	**IDE:** Eclipse, Jdeveloper, Xcode **Version Control:** git, svn **Build Management: -**
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	**Value**
RAM {GB}	4
Disk Storage {GB}	40
vCPUs	1
Network usage {GB}	30000
Hits/Month	15
Registered Users	600
Maximum On-line Users	100
Average On-line Users	50

Blue Parking Valladolid is a client-server application implemented as a centralized service available on Internet and distributed clients running on mobile devices. Server side, the application is deployed on four servers interacting with an SSL certification service, the Spanish national timestamp service and Google Maps. Client side, it interacts with Mapbox and Google Maps. Both the database engine (MongoDB, Oracle SEO) are used in a single deployment for storing different information.

The server side part of the application (candidate to be ported to the cloud) currently implements load balancing and high availability.

The application manages sensitive data and uses domain name *blueparkingvall.com*.

2.1.11 Ayuntamiento de Valladolid - Ideol–Innobarómetro

Ideol–Innobarómetro is an application that allows users to visualize innovative companies. It shows the location where the companies are located within the city area.

The application in currently under development and a prototype (v. 1.0) is available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu
Programming Languages	Javascript, Java, Python, PHP
Databases	PostgreSQL, PostGIS, MySQL
Web/Application Servers	Tomcat, GeoServer, MapServer
Frameworks	OpenLayers
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: svn **Build Management: -**
Open Source Code Repository	N/A

The following table reports the sizing information:

Type	Value
RAM {GB}	4
Disk Storage {GB}	10
vCPUs	2
Network usage {GB}	N/A
Hits/Month	N/A
Registered Users	N/A
Maximum On-line Users	N/A
Average On-line Users	N/A

The application is deployed on a single server and interacts with a Web Map Service (a connection to a map server) and mail servers. The application manages sensitive data like the name of the responsible of the company. It requires a public domain name.

2.1.12 Ayuntamiento de Valladolid – LocalGIS

LocalGIS is a territorial information system for local governments that facilitates municipal management in a georeferenced way, and offers advanced on-line information services to citizens using the cartography of the municipality.

It implements on-line web mapping services, for citizens and city council users and advanced desktop functions for public servants like urban planning, cadastre, inventory, public transportation information, routing, infrastructures, concessions, construction licenses, etc.

It's based on Open GIS technologies (Open Geographic Information Systems).

The application is delivered (v. 2.1) and available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies

Operating Systems	Ubuntu, CentOS, WinServ 2008
Programming Languages	JavaSript, Java
Databases	PostgreSQL, PostGIS
Web/Application Servers	Tomcat, Jetty, MapServer, GeoServer
Frameworks	-
Application Lifecycle Tools	**IDE:** Eclipse **Version Control:** git, svn **Build Management:** Jenkins
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	8
Disk Storage {GB}	200
vCPUs	2
Network usage {GB}	3
Hits/Month	20000
Registered Users	300
Maximum On-line Users	100
Average On-line Users	20

The application is deployed on a single server and interacts with external systems like the Council Web Page, the Document Management System and the Case Management System via web services. It requires a DNS server and implements load balancing (currently implemented with dedicated networking equipment) and high availability (currently implemented with Heartbeat and DRBD).

The application manages sensitive data like personal information of citizens, including cadastre information.

The application does not require a domain name but it's recommended for publishing information to citizens using Web Map Services.

2.1.13 Ayuntamiento de Valladolid - Urbanismo en Red – UeR

Urbanismo en Red (UeR) is created with the purpose of publishing the municipal development plans, across Internet, enabling citizens to access them easily. It is designed for increasing and enhancing transparency in public management of urban sectors. Moreover it provides full interoperability between the various authorities and stakeholders, through electronic services that enable the provision of information of urban planning, to be used by different stakeholders. The application is used by end users (public), operators (GIS technicians) and administrators.

The application is already released (v. 2.0) and is available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu
Programming Languages	Java
Databases	PostgreSQL, PostGIS
Web/Application Servers	JBoss
Frameworks	N/A
Application Lifecycle Tools	**IDE:** N/A **Version Control:** svn **Build Management: -**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	12
Disk Storage {GB}	250
vCPUs	4
Network usage {GB}	2

Hits/Month	50000
Registered Users	N/A
Maximum On-line Users	N/A
Average On-line Users	N/A

The application is deployed on four servers: one application server and one database server both replicated. Application servers require 12 GB of RAM each, while database servers require 6 GB each. Load balancing is implemented using *HAPROXY* and high availability is implemented with *Hearthbeat* and *DRDB*. It requires a domain name: *www10.ava.es*.

2.1.14 URENIO - Virtual City Tour

Virtual City Tour creates an engaging, interactive community map of local sights and attractions. It allows people to discover, in a geographical way, point of interests. For each attraction, it shows a description with useful information and gives the possibility to do a virtual tour to discover in advance the particular point of interest.

The application is delivered (v. 2.0) and available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	Ubuntu, Linux (generic)
Programming Languages	Javascript, PHP
Databases	MySQL
Web/Application Servers	Apache
Frameworks	JQuery, Joomla, Google Maps
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: git **Build Management:**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	**Value**
RAM {GB}	0.5
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	900
Hits/Month	N/A
Registered Users	10000
Maximum On-line Users	10000/day
Average On-line Users	3000/day

This application is deployed on a single server. It requires backup and DNS services.

2.1.15 URENIO - Virtual City Marketplace

Virtual City Marketplace enables the creation of a smart marketplace managed by the local shopping community. It empowers the city local market by bringing together customers and merchants. Local shops are showed inside a local map, along with consumer reviews and promotional offers.

The application is delivered (v. 1.0b) and available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	Ubuntu, Linux (generic)
Programming Languages	Javascript, PHP
Databases	MySQL
Web/Application Servers	Apache
Frameworks	Google Maps

Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: git **Build Management: -**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	0.5
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	450
Hits/Month	N/A
Registered Users	2000
Maximum On-line Users	10000/day
Average On-line Users	1500/day

This application is deployed on a single server. It requires backup and DNS services.

2.1.16 URENIO - Improve My City

Improve My City enables citizens to report local problems such as potholes, illegal trash dumping, faulty street lights, broken tiles on sidewalks and illegal advertising boards. The submitted issues are displayed on the city map. Users may add photos and comments. Moreover, they can suggest solutions for improving the environment of their neighbourhood. The application is also available as a mobile app for Android and iPhone devices. The mobile app is fully interconnected with the web platform and supports the full range of the aforementioned features adding some extra functionality based on the capabilities of modern smartphones.

The application is delivered (v. 2.5.7) and available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu, Linux (generic)
Programming Languages	Javascript, PHP
Databases	MySQL
Web/Application Servers	Apache
Frameworks	JQuery, Joomla, Google Maps
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: git **Build Management: -**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	0.5
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	75
Hits/Month	N/A
Registered Users	20000
Maximum On-line Users	1500
Average On-line Users	500/day

This application is deployed on a single server. It requires backup and DNS services.

2.1.17 URENIO - Sense the City

Sense the City is an open source web application that receives and visualizes air pollution data from sensors around the city. It currently supports Libelium devices (see http://www.libelium.com), giving the possibility to interfaces with a large amount of sensor device models.

The application is delivered (v. 1) and available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu, Linux (generic)
Programming Languages	Javascript, PHP
Databases	MySQL
Web/Application Servers	Apache
Frameworks	JQuery, Joomla, Google Maps, NodeJS
Application Lifecycle Tools	**IDE**: Eclipse **Version Control**: git **Build Management:**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	0.5
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	60
Hits/Month	N/A
Registered Users	10000
Maximum On-line Users	10000/day
Average On-line Users	200/day

This application is deployed on a single server. It requires backup and DNS services.

2.1.18 URENIO - Honolulu Answers

Honolulu Answers is a new approach to make it easier for people to navigate city information and services quickly. It's a citizen-focused website that is question-driven, with clean, easy-to-navigate design. Unlike a portal destination, Honolulu Answers is like Google -- type in anything, and it probably gives you the answer you're looking for, using the words you know. Every page on the site is an answer to a potential Google search question by a citizen, written in simple, friendly language, as if you'd asked your neighbor a question. The content is organized based on citizen understanding, the intuitive way you'd think of a problem, not the way the city is organized internally.

The application is delivered here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu
Programming Languages	Ruby
Databases	PostgreSQL
Web/Application Servers	N/A
Frameworks	N/A
Application Lifecycle Tools	**IDE**: N/A **Version Control**: git **Build Management:**
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	1G
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	150

Hits/Month	N/A
Registered Users	100
Maximum On-line Users	5000/day
Average On-line Users	1000/day

This application is deployed on a single server.

2.1.19 URENIO – OpenTripPlanner

OpenTripPlanner is an open source multi-modal trip planner. It depends on open data in open standard file formats (GTFS and OpenStreetMap), and includes a REST API for journey planning as well as several map-based Javascript clients. OpenTripPlanner can also create travel time contour visualizations and compute accessibility indicators for planning and research applications.

The application is delivered (v. 0.9.1) and available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	N/A
Programming Languages	Java
Databases	N/A
Web/Application Servers	Tomcat, Geoserver
Frameworks	-
Application Lifecycle Tools	**IDE:** Eclipse, NetBeans **Version Control:** git **Build Management:** N/A
Open Source Code Repository	Link

The following table reports the sizing information:

Type	**Value**
RAM {GB}	1
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	240
Hits/Month	N/A
Registered Users	0
Maximum On-line Users	2000/day
Average On-line Users	800/day

This application is deployed on a single server. It requires backup and DNS services.

2.1.20 URENIO – Crowdtilt

Crowdtilt is a full-featured, open-source, customizable crowd-funding tool that allows anyone to launch their own campaign. It is a powerful and flexible crowd-funding solution for brands, businesses and organizations.

The application is delivered (v. 2.0) and available here.

Technical Information

The application is implemented using the following technologies:

Type	**Technologies**
Operating Systems	N/A
Programming Languages	Ruby
Databases	PostgreSQL
Web/Application Servers	Apache, nginx
Frameworks	RubyOnRails, ImageMagick
Application Lifecycle Tools	**IDE:** N/A **Version Control:** git **Build Management:** N/A
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	1
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	150
Hits/Month	N/A
Registered Users	5000
Maximum On-line Users	1000/day
Average On-line Users	500/day

This application is deployed on a single server.

2.1.21 URENIO – LocalWiki

LocalWiki is tool for collaborating in local, geographic communities. It supports a grassroots effort to collect, share and open the world's local knowledge.

The application is released (v. 0.5.5) and available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu, CentOS
Programming Languages	Python
Databases	PostgreSQL, PostGIS
Web/Application Servers	Apache
Frameworks	Django, Cloudmade
Application Lifecycle Tools	**IDE:** **Version Control**: git **Build Management:** N/A
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	0.5
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	240
Hits/Month	N/A
Registered Users	10000
Maximum On-line Users	2000/day
Average On-line Users	800/day

This application is deployed on a single server. It requires backup and DNS services.

2.1.22 URENIO – OpenCivic

OpenCivic is a distribution of Drupal (https://drupal.org/) designed to support communities of software developers in creating, cataloguing and sharing software applications. It is based on Code for America's Civic Commons project, which was created as a platform for sharing information specifically about "civic software" used by governments and nonprofit organizations to provide public services. The main goal of this distribution is to help build websites that enable people to share information about software applications.

The application is released and accessible here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu, Linux (generic)
Programming Languages	Javasrcipt, PHP

Databases	MySQL
Web/Application Servers	Apache, nginx
Frameworks	Drupal
Application Lifecycle Tools	**IDE:** N/A **Version Control:** git **Build Management:** N/A
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	1
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	30
Hits/Month	N/A
Registered Users	1000
Maximum On-line Users	1000/day
Average On-line Users	200/day

This application is deployed on a single server. It requires DNS services. In addition to the standard Drupal requirements it needs the following Drupal add-ons: *drush 3.1* and *drush make 2.0 beta 9*.

2.1.23 URENIO - We the People Petitions

We The People Petitions gives citizens a way to create and sign petitions on a range of issues affecting their city. It is a Drupal 7 code base used to build an application that lets users create and sign petitions. It's based on an initiative of President Obama's government to create the most open and participatory government of US history, and this petitioning platform is a key part of that initiative. Site visitors can create a user account, log in, and create petitions. Petition creators can share the URL for their petition to generate signatures. When the petition crosses a certain threshold, the petition becomes "public" and is listed as an open petition on the site's "open petitions" page. Visitors can sign petitions. Petitions that receive a designated number of signatures (at the White House the number is 100,000 in one month) get a response.

The application is released (v. 2.0) and available here.

Technical Information

The application is implemented using the following technologies:

Type	Technologies
Operating Systems	Ubuntu
Programming Languages	Javasrcipt, PHP
Databases	MySQL, MongoDB
Web/Application Servers	N/A
Frameworks	Drupal
Application Lifecycle Tools	**IDE:** N/A **Version Control:** git **Build Management:** N/A
Open Source Code Repository	Link

The following table reports the sizing information:

Type	Value
RAM {GB}	1
Disk Storage {GB}	10
vCPUs	1
Network usage {GB}	450
Hits/Month	N/A
Registered Users	20000
Maximum On-line Users	5000/day
Average On-line Users	1000/day

This application is deployed on a single server. It requires DNS services. Although not strictly required, a dedicated MongoDB server is recommended.

2.1.24 RTDI - PLAY and STREAM

Play and *Stream* are two candidate objects that need to be dealt separately. Actually, they aren't applications; instead they are technologies that could be used in the implementation of the cloud platform architecture.

PLAY

PLAY develops and validates an elastic and reliable architecture for dynamic and complex, event-driven interaction in large highly distributed and heterogeneous service systems. Such architecture will enable ubiquitous exchange of information between heterogeneous services, providing the possibilities to adapt and personalize their execution, resulting in the so-called situational-driven process adaptivity. It is implemented using the *Java* programming language and uses *MongoDB* as database. It is deployed to the *Apache Tomcat* application server. The source code is available here.

STREAM

STREAM is an architecture that provides real-time services over massive data flows. The main characteristics are:

- *Scalability*: Scaling in the data stream volume in addition to scaling in the number of queries and/or operators and able to scale to 100s of nodes.
- *Elasticity*: Growing and shrinking the number of nodes as needed to cope with the incoming load and minimizing the used resources.
- *Low latency and high throughput* network communication.
- High throughput storage able to store *streaming data at network rates*.

2.2 Assessment Results

This section discusses the information gathered during the application assessment. Details for each application are provided in the previous section, now the goal is to wrap up and come up with a list of common functions and technologies used for implementing the applications. They will influence the design of the cloud platform in order to facilitate the transportation.

Most of the applications are implemented using open source products at different levels (e.g. operating systems, databases, frameworks, developing tools and web servers). The use of open source software is crucial both in general and specifically for this project because European Commission (EC) "*supports Free/Open Source Software (FOSS) as a development model since it is a very effective way to collaboratively develop software with fast take-up and improvement cycles. It is more and more used as a vehicle for the dissemination of results of ICT research projects. Those aspects are particularly important for the development of the information society, and in particular of the future internet of services, as FOSS provides open and adaptable building blocks that lower the barriers to entry to new service providers and allow them to develop and innovate faster*" [4]. The EC developed a strategy to use even in their internal systems a family of OSS products and tools.
On the other hand, few applications are implemented with proprietary technologies (e.g. Oracle DBMS, Windows Server 2008, etc.) raising issues when porting applications to cloud. Problems are related to the licensing of the software products used for implementing the applications: some software vendors base their licensing on the number of users (named or concurrent), others charge on a per-processor or per-core basis still other vendors define some 'exotic' different usage metric. When porting an application to cloud, these aspects need to be thoroughly examined in order to be sure that the act of "porting the application to cloud" doesn't constitute an infringement of the licensing rights that the application proponent(s) have in place with the software vendor. As a general rule, any right not explicitly stated as being granted to the customer in the license agreement is retained by the software manufacturer.
A full analysis of the issues raised by the use of licensed software products in the application is largely out of the scope of this document and the Software Cloud Platform design, in general.
It is worth noticing that the usage of proprietary software solutions could also constitute an exception to the project objectives as reported in the DOW: "*No proprietary approach is envisaged during the project. The project partners declare strong commitment to an 'open' approach. This is manifested by: (i) releasing the developed technologies under OSS (open source software) (ii) relying on open innovation methodologies; (iii) promoting open standards, (iv) putting emphasis on usability and reusability and interoperability of technological solutions deployed under the project*", [3].

In addition, the applications based on licensed software could be less attractive for organizations willing to reuse the implemented functions because of the related license cost. This could 'frustrate' the inclusion of such applications in the catalogue of the reusable applications.

The analysis of the technical aspects of the applications shows that most of them are implemented as LAMP software bundles.
According to a broadly accepted definition [5], *"The acronym LAMP refers to first letters of the four components of a solution stack, composed entirely of free and open-source software, suitable for building high-availability heavy-duty dynamic web sites, and capable of serving tens of thousands of requests simultaneously".*

The meaning of the LAMP acronym depends on which specific components are used as part of the actual bundle, in this case:

- ***L**inux, the operating system (i.e. not just the Linux kernel, but also glibc and some other essential components of an operating system)*
- ***A**pache HTTP Server, the web server*
- ***M**ySQL, **MariaDB** or **M**ongoDB, the database management system*
- ***P**HP or **P**ython, the scripting languages (respectively programming languages) used for dynamic web pages and web development.*

Many candidate applications are developed and built using these technologies, which are considered standard de facto.
There are some notable variations to this baseline. Some applications require the PostgreSQL as relational database management system instead of one of above mentioned DBMS but this is not a problem, since PostgreSQL is open source, wide used and supported. Other applications are implemented with different languages (e.g. Java or JavaScript) or are based on different frameworks (NodeJS, ExtJS, JQuery, Joomla, Drupal).
An interesting common aspect of the applications is the type of workload; almost all the applications have significant need for web and/or mobile access and this is an important point to take into account when designing the proper cloud solutions. As a matter of fact, the market offers several cloud solutions specifically designed for supporting this type of workload. Some solutions are available as 'pure services' meaning that users can only purchase the ability of running applications in a proprietary cloud platform made available via Internet. On the other hand, there are solutions that can be implemented on-premise by installing on in-house hardware proper software solutions providing similar services. Software platforms like OpenShift and CloudFoundry are open software Platform as a Service (PaaS) software solutions specifically designed for supporting web-based applications. It is worth noticing that these platforms significantly facilitate the deployment and the management of applications in the cloud and, in some cases, can also simplify the development by integrating software versioning and software repository systems (e.g. SVN and Git), software Integrated Development Environments (e.g. Eclipse).
The Storm Cloud Platform design shall take into account the findings of the application service assessment and investigate the possibility of using open software PaaS solutions.

The assessment shows that some applications manage sensitive information like, for example, personal data of the users.
This is a critical aspect for cloud computing in general and for Storm Clouds project in particular.
Beside the implementation of extensive security controls like unauthorized access prevention, data encryption and ad hoc firewall policy, there still remain questions where data are located. In fact, data is stored physically in a particular country and is subject to local rules and regulations. Using applications that manage sensitive data implies, for the cloud platform, to be aware of such rules and regulations and acts in order to respect it: for example to store data in a particular data centre located in a particular region. Such actions could complicate, or even inhibit the porting of applications to cloud in particular if "public cloud" services are used. This aspect needs to be thoroughly examined during the project in particular for the application portfolio definition because it can be a critical aspect for the adoption of the services.

3 Storm Clouds Platform Requirements and Specification

This chapter defines the Storm Clouds Platform requirements and specification.

3.1 The NIST Definition of Cloud Computing

The National Institute of Standard and Technology (NIST) provides a definition of cloud computing along with the definition of some important related aspects [6]. These definitions are intended "*to provide a baseline for discussion from what is cloud computing to how to best use cloud computing*".

Cloud computing is a complex topic spanning a wide spectrum of underlying technologies, configuration possibilities, provided services and deployment models. NIST gives a general, yet precise, classification of the various aspects to consider when defining or analyzing cloud computing, in general, or specific cloud computing systems.

In this paragraph, the NIST definitions are analyzed and challenged with the objective of eliciting requirements applicable to Storm Clouds Project and the implementation of SCP and SCPP.

3.1.1 Cloud Computing Definition

NIST defines cloud computing paradigm as follows [6]:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

The definition can be illustrated by the following picture:

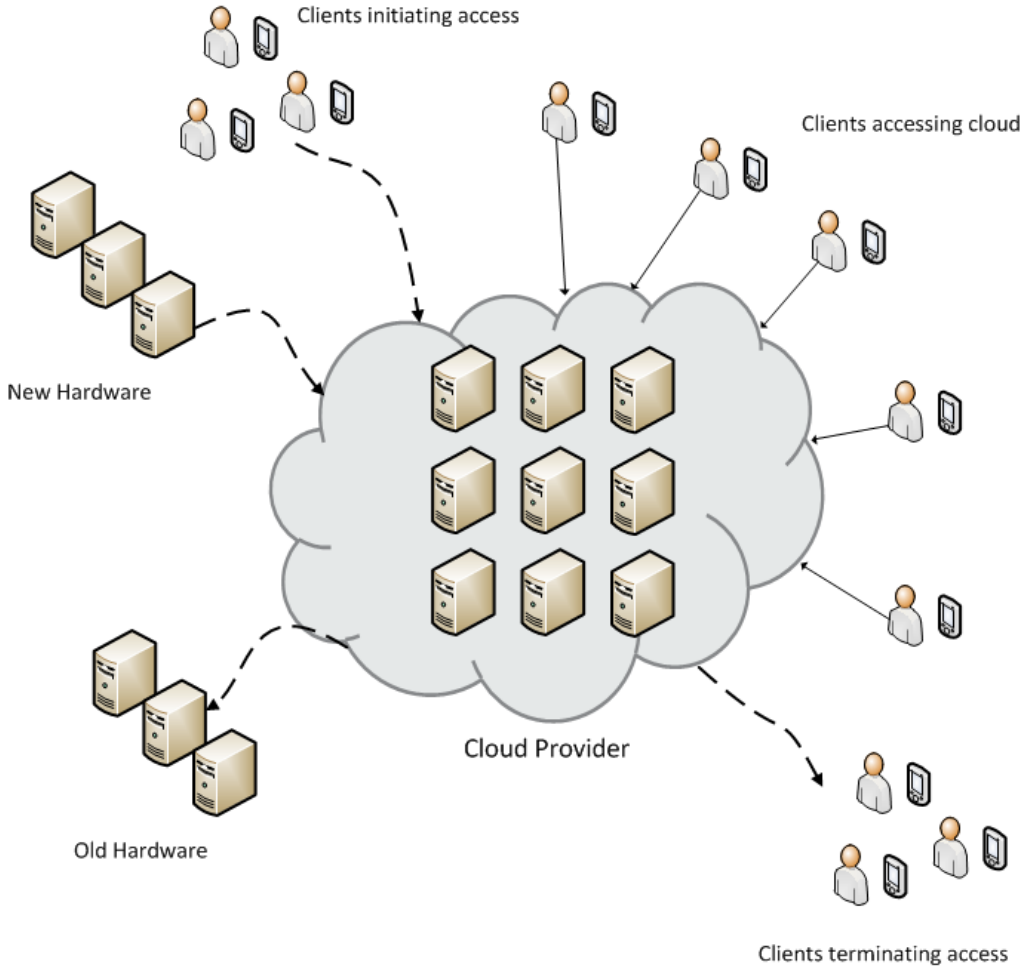

Figure 3-1 – Cloud Computing Definition

The figure shows users that, using a **client** technology, access via a network connection a set of cloud services made available by a **cloud provider**. Users – also called **cloud consumer** – may arrive and depart making the use of cloud services variable. The cloud provider implements cloud services using a set of physical hardware resources (i.e. computer servers, disk storage and network equipments/connections) that can be

added and retired at any time (from the cloud consumer viewpoint), in order to meet a level of service the provider is wanting or is required to implement.

In the paragraph above, some terms have been introduced and need a formal definition NIST provides [7]:

- ***cloud consumer*** *or* ***customer****: a person or organization that is a customer of a cloud; note that a cloud customer may itself be a cloud and that clouds may offer services to one another;*
- ***client****: a machine or software application that accesses a cloud over a network connection, perhaps on behalf of a consumer*;
- ***cloud provider*** *or* ***provider****: an organization that provides cloud services.*

In the Storm Clouds Project perspective, cloud computing model and the explanation above elicit some considerations that need to be taken into account.

The SCP design, for example, shall implement the possibility of accessing the cloud services it implements via a network connection and, more specifically, SCPP (the actual SCP deployment implemented for shared usage) shall be reachable via Internet. This is a direct consequence of the project consortium that involves organizations that are located in different countries making it impossible to allow personnel to access a location where the actual cloud infrastructure is installed (i.e. data centre).

NIST definition claims that the access to resources is to be convenient. This assertion can be interpreted from different angles. From the cloud consumer view point it could be related to the price s/he pays for a subscription to cloud services. In the project context, this doesn't appear to have much importance. From the cloud provider view point, on the other hand, the idea of convenience could be related to an efficient use of the physical resources (e.g. servers) used for implementing an actual platform. In the project context, this is important aspect for controlling the cost of the SCPP according to the budget. In more general terms, a SCP design aimed to efficiently use computing resources can be beneficial also for those project participants willing to implement their own cloud. The application assessment provides interesting indications on this topic. All the analyzed applications can be classified as "web-based" and (part of) their workload present characteristics of that type. Based on this information, SCP design could include solutions specifically designed for this kind of workload/application that, at the same time, facilitates the migration to cloud of the applications and optimize the resource usage

Figure 3-1 – Cloud Computing Definition shows that the cloud provider can add and remove hardware to the physical infrastructure. This aspect has several implications on actual cloud computing platform in general and on SCP and SCPP in particular.
Generally speaking, the removal of hardware (e.g. servers) has the direct consequence that the workload currently running on it must be moved to another device in order to meet business continuity requirements. In this perspective, SCP and – possibly - SCPP shall implement this sort of function at least for some low level cloud service it implements (i.e. IaaS level). Mechanisms like virtual machine live migration can be handy for this purpose.

Another interesting implication, more related with the project, is the ability to incorporate new hardware to an already existing infrastructure. In general, cloud providers add new hardware to their infrastructure both for substituting removed equipment but also for meeting requirements of higher computing power and this is certainly a fundamental aspect that SCP must cover. On the other hand, from the project perspective, although we cannot *a-priori* exclude the former reason (maybe for substituting failing equipments), the latter is certainly worth exploring for eliciting requirements for SCPP.
According to the project plan, the Storm Clouds Project is based on a methodological framework "*[…] divided in 2 stages, involving all stakeholders and followed by the Pilots in the project*. More in detail: "*During the first stage, they (*i.e. project participants*) will be working in the Cloudification of services (Users and stakeholders' involvement, development and deployment of services) and validation with users. During the second stage, pilots partners will be working as replication pilots, transferring services of their interest that were cloudified in the first stage by another Lead Pilot. Pilots will be running then, for 4 Innovation Cycles of 9 months composing 2 stages of 18 months each*" [3].
We can imagine that the workload for the SCPP will increase according to the project evolution; it is therefore sensible to allocate fewer physical resources in the first stage and add further physical resources later. Then, SCP is to be designed for this evolution and allow hardware addition without requiring full re-installation re-configuration of the software.

According to *Figure 3-1 – Cloud Computing Definition*, using cloud services, cloud consumers share computing resources; the kind of resources they share strongly depends on the cloud services they are using as well as on the implementation of the services put in place by the cloud provider. Servers can be shared using hypervisor

based technologies or operating system-level virtualization method (e.g. LXC, see https://linuxcontainers.org/), disk space can be shared using virtualization technologies like iSCSI and physical network resources can be shared using Virtual Private Network (VPN) or tunnelling technologies (e.g. GRE tunnelling). Generally speaking, in a cloud implementation, a fundamental issue is how and to what extent to support concurrent usage of the same physical resources and, at the same time, ensure that two different cloud users do not incidentally or maliciously interfere each other compromising the security of the hosted applications and/or data. For this purpose, cloud systems implement multi-tenancy that can be defined as "*an architecture in which a single instance of a software application serves multiple customers. Each customer is called a tenant*" [8]. According to another but similar definition multi-tenancy is "*a reference to the mode of operation of software where multiple independent instances of one or multiple applications operate in a shared environment. The instances (tenants) are logically isolated, but physically integrated*" [9].
From the project perspective, Storm Clouds Project participants might have different objectives while using cloud computing and the applications they propose for being ported onto a cloud platform could require some sort of "isolation" from the others in order to meet security and/or privacy requirements. For these reasons, SCP should be designed for supporting multi-tenancy and SCPP should take advantage of that for providing project participants with one or more tenants they can use for hosting their applications. Cloud solutions take advantage of multi-tenancy also for limiting the amount of resources a cloud consumer is allowed to use. Storm Clouds Project and, more specifically, SCPP can take advantage of this feature for optimizing the overall usage of the physical resources.

3.1.2 Essential Characteristics

NIST defines essential characteristics of cloud computing paradigm but not all the actual cloud implementations require or simply can afford their full realization. For example, 'rapid elasticity', for an organization providing public cloud services, might be a must but, for an organization implementing an on-premises cloud, the allocation of resources for a certain workload could require the validation of an operator in charge of administering and supervising the usage of the whole underlying infrastructure.

The essential characteristics are reported in the following table along with considerations from the Storm Clouds Project perspective.

NIST Definition	Storm Clouds Project Considerations
***On-demand self-service**: a consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.*	Some Storm Clouds Project's participants are in charge of adapting applications for running on top of SCP. In order to streamline the process, SCP shall implement a web-based user interface and an API that allows them to autonomously allocate basic computational resources like virtual machines and virtual storage disks for installing, adapt, deploy and run applications.
***Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).*	From the project viewpoint, it is important to estimate the bandwidth required for supporting the applications in the SCPP and make sure that the actual implementation takes into account both the resource required and the project budget limitations. In addition, albeit the technology used for accessing applications strictly depends on their specific implementation, SCP shall not prevent the development/deployment of applications supporting heterogeneous clients and protocols (multichannel should be supported).
***Resource pooling**: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.*	For what concerns multi-tenancy, the aspect has already been covered above. Location independence, also called location transparency, needs some analysis. The application assessment shows that some applications manage sensitive information like, for example, data regarding individuals. The European Community have promulgated regulations aimed to protect individuals with regard to the processing of personal data and on the free movement of such data [10]. These two aspects need to be taken into

NIST Definition	Storm Clouds Project Considerations
	account during the project. For what concerns SCPP, a direct consequence is that the infrastructure, used for implementing the cloud, is provided by an entity (the infrastructure provider) that is compliant to the above mentioned directive. This implies, for example, that the data centre is actually located in an EU country. Nonetheless, this might be not sufficient; in some cases candidate applications could be judged as "not SCPP deployable" meaning that SCPP cannot guarantee the security level required for the information they manage. As far as SCP is concerned, it should implement mechanisms for providing cloud users with some control on the location of computing resources and/or data. As a matter of fact, we can imagine situations where a project participant (e.g. a municipality) implements its own SCP and wants to provide cloud users with the ability to control such an aspect for implementing a certain policy. For example cloud users might be provided with the ability to require that "this file shall not leave the computers deployed in this office". Such a future should be implementable by SCP albeit it requires that the cloud provider (in this case the municipality) correctly deploys and configure SCP.
***Rapid elasticity**: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.*	This is one of the most attractive promise of cloud computing and, to some extent, is also implemented (at least from the cloud consumer perspective) when a cloud consumer uses a public cloud (e.g. Amazon Web Service or HP Cloud Service) For what concerns the project, the expectation should be more realistic. SCPP, for example, will be implemented taking into account the budget limitations of the project and rapid elasticity won't be implemented. On the other hand, SCP could be designed using cloud solutions that permit rapid elasticity, to some extent. In particular, SCP could provide cloud consumers with functions for allocating basic computing resources (e.g. virtual machines) and they should be provided both via a web-based UI and an API. The latter opens the possibility of implementing automatic provisioning of resources.
***Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.*	Generally speaking, metrics are usually collected for resource usage monitoring, capacity planning, and service usage billing purposes. For what concerns SCPP, resource usage monitoring is applicable in order to highlight overloading situations and allow the cloud provider to take some corrective action. This can result with the optimization of the available resources re-allocating resources to tenants and, provided that it is permitted by the project budget limitations, add new physical resources to the platform. This implies that SCP implements some basic measurement functions aimed to monitor the usage of physical and/or virtual resources (e.g. VCPUs, RAM, disk space, network traffic).

Table 3-1 – Cloud Computing Essential Characteristics

3.1.3 Deployment Models

Cloud computing paradigm, by its very nature, introduces issues regarding the control cloud consumers have on the computing resources they use. In fact, in cloud computing, consumers put their workload and their data 'in the hands' of the cloud provider who provides the computing resources for managing them. By doing that, cloud consumers give up to cloud providers both **control** (the ability to **decide** who and what can access data and programs) and **visibility** (the ability to **monitor** that status of the resources including the access to them by other users or programs).

NIST defines deployment models for describing to what extent a cloud user relinquishes control and visibility to the cloud provider s/he selects; there are four deployment models (private, community, public and hybrid) with relevant variants related to the location where the physical infrastructure is deployed (private on-site and private outsourced, community on-site and community outsourced).

Deployment models permit an increasing level of control and visibility as shown in the figure below[2]:

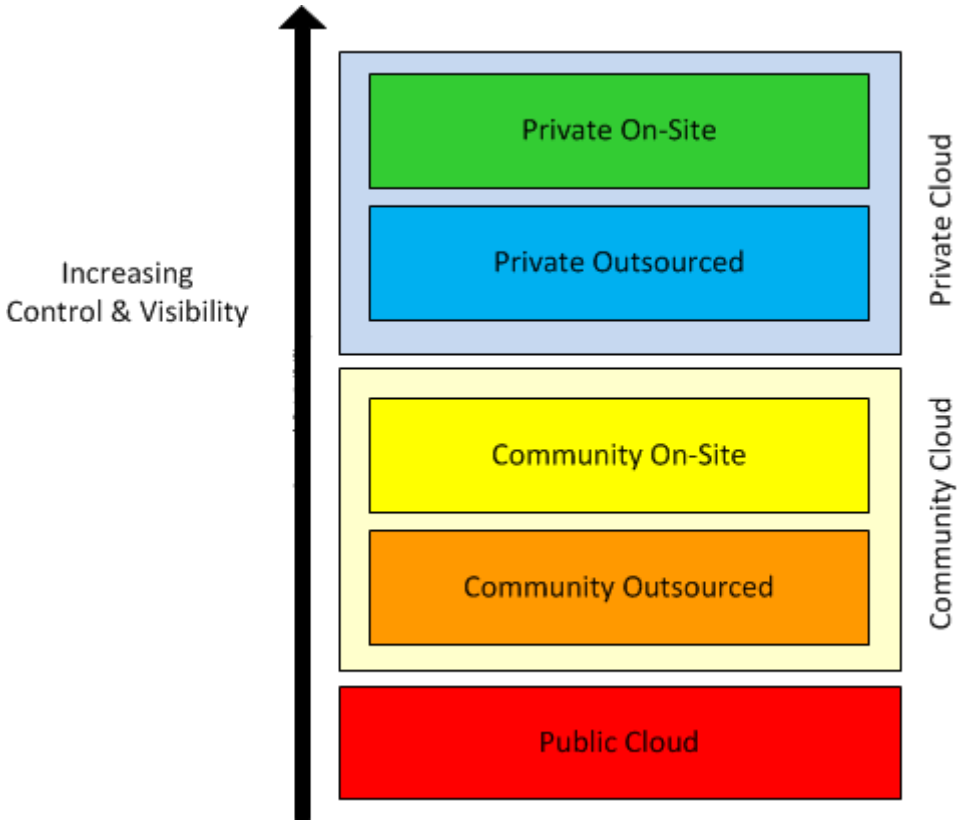

Figure 3-2 – Cloud Computing Deployment Model – Control and Visibility

It is important saying that the actual level of control and visibility does not solely depend on the deployment model a cloud consumer decides to adopt; it also depends on the policies and the processes the consumer and/or the provider put in place. For example, choosing a private on-site cloud, a cloud consumer is in the best position for implementing very strict access control policies because s/he (or the organization s/he belongs to) physically owns the hardware infrastructure where the cloud is implemented. However it's up to the consumer, and/or the organization s/he belongs, to ensure that the needed access control policies are actually implemented. To summarize, a deployment model, in itself, does only define **a level of confidence** a consumer have on control and visibility on the access to the data and programs s/he puts into the cloud.

The deployment models are reported in the following table along with considerations from the Storm Clouds Project perspective.

NIST Definition	Storm Clouds Project Considerations
Private cloud. *The cloud infrastructure is provisioned*	As mentioned above, there are two variants of this

[2] Hybrid cloud, being a combination of all the other models, is not reported.

NIST Definition	Storm Clouds Project Considerations
for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.	model that depend on the organization in charge of providing the physical resources for hosting the cloud implementation. In on-site private scenarios, the organization providing the hardware resources is the same the cloud consumer belongs to; in outsourced private scenarios the cloud infrastructure is provided by an external entity. The consumer still preserves a very high level of control because the infrastructure is dedicated to implement a cloud for exclusive use of the consumer's organization. In addition, in outsourced private scenarios, the consumer's organization can require the infrastructure provider puts in place processes and technical solutions for enforcing policies that the provider must respect. In the project context, private cloud is applicable, to some extent. As described in [3], some project participants may be willing to deploy services on equipment at their own sites, maybe because of the kind of data managed by the applications that require particular security measures. In such a case, SCP design shall support the private (on-site) deployment model.
Community Cloud. *The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.*	In community cloud, members of a community share their resources for implementing a cloud that is accessed and used by all the community participants for some common objective. Each participant may provide cloud services, consume cloud services or both. Similarly to the private cloud case, community cloud have two variants on-site community cloud, where participant physically owns the hardware on their sites, and outsourced community cloud, where participants lease the cloud infrastructure from a provider. Similarly to outsourced private cloud, also in outsourced community cloud consumers (in this case are also participants of the community) can require the cloud provider to implement some policies for increasing the level of control and/or vision they have on the cloud resources. In the project context, SCPP is a specific case of outsourced cloud community: Storm Clouds consortium is a community, consortium partners are the community members, SCPP is a cloud providing services for them to use. As reported in the [3], during the project, the consortium "*will define and implement common infrastructural solutions and components for deploying cloud-based services [...omitted...] possibly leasing computational resources at a public cloud provider*". In such a case, the public cloud provider is an entity external to the community; therefore the infrastructure is outsourced.
Public cloud*. The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.*	A public cloud is a model in which a service provider makes cloud services available to the general public over the Internet. Actual examples of public clouds are Amazon Web Services (http://aws.amazon.com), Hewlett Packard Cloud Services (http://www.hpcloud.com) and RackSpace Public Cloud (http://www.rackspace.com/cloud). A cloud consumer, using cloud services provided by a public cloud, has generally very little control and visibility

NIST Definition	Storm Clouds Project Considerations
	on how data are preserved and processing are accomplished but, in return, has the illusion of unlimited resource availability because public cloud operators make huge investments on hardware equipment for supporting heavy workloads. In the project context, as already mentioned, public cloud services could be an option for implementing SCPP but the cloud services used by project participants will be offered by SCPP, not directly by the underlying infrastructure leased form a public cloud provider. As an additional consideration, it would be beneficial for the applications adapted to cloud during the project, that the technologies and the cloud solutions used for implementing SCP are also available in the public cloud market. In such a case, the applications can be easily migrated to a public cloud once the project terminates and, consequently, the SCPP is dismantled. The effort for adapting is saved to some extent.
***Hybrid cloud**. The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).*	In hybrid cloud scenarios, the cloud is a composition of two or more clouds (private, community, or public) bound together by data and application portability. For what concerns the project, a realistic scenario to explore is the possibility of deploying in the cloud (SCPP or a private SCP deployment) only parts of an application in order to benefit of the cloud strengths (e.g. scalability) without jeopardizing other important aspects (e.g. security) For example, a two layer application with a web front-end and a database keeping sensitive information could be deployed partially in the cloud (i.e. the web front-end in the cloud) and partially on-site (i.e. the database in the local traditional infrastructure or in an on-site SCP private cloud).

Table 3-2 – Cloud Computing Deployment Models

3.1.4 Service Models

A cloud computing system provides access to different services that range from software applications, such as email or productivity tools that can be directly used by end-users (SaaS), environments for building, deploying and operating software applications (PaaS); and traditional computing resources, such as processing power, storage and network connectivity (IaaS).

The deployment models that NIST defines are reported in the following table along with some explanations and considerations from the Storm Clouds Project perspective.

NIST Definition	Storm Clouds Project Considerations
***Software as a Service (SaaS)**. The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings*	A cloud SaaS can be described as an application, running on a cloud infrastructure, accessed over a network (e.g. Internet) and implementing some specific functions for cloud consumers. Examples of SaaS are messaging systems (e.g. web-based mailing systems), web-based office suites (e.g. Google docs), customer relationship management systems, application lifecycle management systems (e.g. HP Application Lifecycle Management), etc. The novel aspects of a cloud SaaS in respect to a 'traditional' service available via Internet and providing similar functionally are that users are charged on a per-usage basis and that the computing resources used

NIST Definition	Storm Clouds Project Considerations
	for running the application are allocated on an 'on-demand' basis. Usually, SaaS applications are designed for being highly scalable and take advantage of the underlying resources that are made available following a cloud computing paradigm. For example, a SaaS application providing web-office suite functionality could store files into an Object Storage service that is one of the services provided by an IaaS cloud. In such a case, a SaaS application shall explicitly use an IaaS cloud service for the implementation but this relationship usually results completely transparent to end-users. Application development is neither in the SCP nor SCPP scope. As a matter of fact, SCP is designed for hosting applications provided by project stakeholders but it does not implement any cloud SaaS in itself. Applications might require some sort of adaptation for running on top of SCP and can also take advantage of SCP services (either IaaS or PaaS) for implementing some feature. For example, an application might require the implementation of back-up of the data they manage; SCP, providing Object Storage services (a IaaS level service), can facilitate the implementation of such a feature with a suitable API that is called by the application itself. In addition, adapted applications will be consolidated in a service portfolio that contains "*cloud-based public services that could be transferred to other cities not taking part in the project and/or scaled up to wider geographical scopes*" [3]. SCP shall support the portfolio with an on-line catalogue containing templates for facilitating the adaptation and deployment of cloud-based applications.
***Platform as a Service (PaaS)**. The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.*	A cloud PaaS implements services for facilitating the development, testing, deploying and administration of applications running in a cloud environment. Applications are actually hosted in a framework that provides tools supporting the developers and administrators in any phase of the application lifecycle without requiring an explicit responsibility on the management and administration of the underlying computing infrastructure (either physical or virtual). Using a PaaS, an application developer can focus on the construction of the application software and does not take care of low level aspects related to the machine where it eventually operates. A cloud PaaS is usually characterized by the computing languages, the libraries and frameworks, the run-time environments and the development tools it implements or supports. PaaS solutions are usually specifically designed for supporting a certain kind of workload: there are PaaS solutions for implementing web applications (e.g. OpenShift and CloudFoundry), solutions for big-data applications (Savanna), solutions for grid-computing applications, etc. From the application developer view point, the deployment of an application in a PaaS cloud is as easy as uploading a file containing all the software artefacts that build-up the application to the PaaS cloud. The

NIST Definition	Storm Clouds Project Considerations
	cloud provider, on the other hand, can receive benefits from using a PaaS cloud solution because usually it makes an efficient usage of the physical resources where applications eventually run. As a matter of fact, some PaaS cloud solutions can host more than one application in a single (virtual) machine implementing a sort of separated computing environment at the same time. In the project perspective, the application assessment shows that all the candidate applications can be classified as web-based or, at least, have some components that can be classified as such. In addition, most of them are constructed using standard and open technologies supported by PaaS software platform solutions available under open source licensing (e.g. OpenShift, CloudFoudry). For those reasons it makes sense to require that SCP implements a sort of PaaS cloud solution in order to facilitate both the developers and the administrator tasks.
Infrastructure as a Service (IaaS). *The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).*	An IaaS platform is a software layer that provides services for creating virtual resources like virtual machines, virtual disks and virtual networks. They can be used instead of their physical counterparts in order to deploy and run applications. Virtual resources are provided as services meaning that they are created when needed, used to run applications and destroyed when the application is no more needed. The actual computation happens at the physical level but physical resources and applications are not tightly bound each other. This makes it easier to reuse the physical infrastructure for several purposes usually at different times. The application assessment shows that many different technologies have been used for implementing the candidate applications. There are different Operating Systems (e.g. Windows and some Linux distributions), different languages (e.g. Java, Ruby, PHP, Python), different database engines (e.g. MySQL, PostgreSQL, Oracle, etc.), different frameworks and/or libraries (e.g. PostGIS), etc; in addition, some of the technologies are proprietary (e.g. Windows and Oracle). At the time being, there's no PaaS solution covering all the technologies used for implementing all the candidate applications, therefore – in order to have the ability to support most of the candidate applications – SCP should implement IaaS services that provide more flexibility than a certain PaaS platform.

Table 3-3 – Cloud Computing Service Types

Applications are implemented as a stack of software and hardware layers; the following list enumerates and briefly describes the layers:

- **Application Layer**: it contains the logic that implements the functions delivered to the end-users (e.g. a mail messenger implements functions for sending e-mail, administering accounts, etc.);
- **Middleware Layer**: it contains software tools used for implementing the application (e.g. programming languages, database engines, frameworks, etc.);

- **Operating System**: it is the collection of software that manages (virtualized) hardware resources and provides common services to computer programs;
- **Virtualization Layer**: it contains base software for implementing virtual hardware resources (e.g. virtual machines, virtual disks, virtual network objects);
- **Hardware Layer**: it contains the physical equipment for hosting the cloud platform implementing cloud services as well as the applications running on top of it.

It's worth noticing that in this context Virtualization Layer contains both an operating system, running directly on top of the physical hardware, and virtualization software (e.g. hypervisor), that "creates the illusion" of having more physical equipments than those actually available at the physical level . When the virtualization layer is implemented (this is required for realizing a cloud IaaS but it's optional for a cloud PaaS or a cloud SaaS), the stack includes more than one operating systems: one (called host operating system) running on the physical hardware and the others (called guest operating systems) running in the hypervisor context:

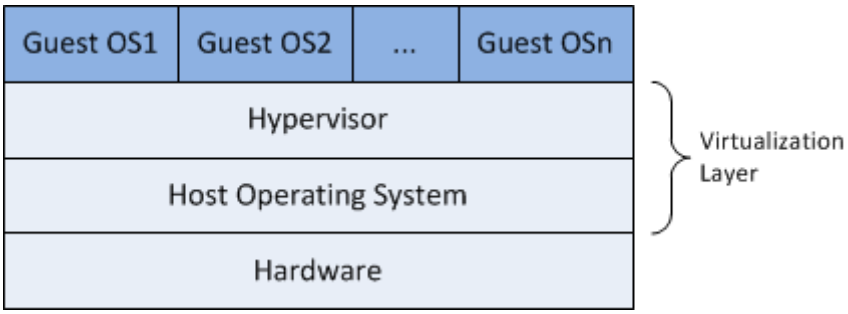

Figure 3-3 – Virtualization Layer

Depending on the cloud service type, cloud consumers and cloud administrators exercise different level of control on the layers, as reported in the following table:

Type	Layer	Consumer	Provider
SaaS	Application	Limited Admin & User Control	Admin Control
	Middleware	No Control	Full Control
	Virtualization (opt.)		
	Operating System		
	Hardware		
PaaS	Application	Admin Control	No control
	Middleware	Used as a service for building applicatios	Admin Control
	Operating System	No control	Full control
	Virtualization (opt.)		
	Hardware		
IaaS	Application	Full control	No control
	Middleware		
	Operating System		
	Virtualization	Used as a service for activating virtual resources	Admin control
	Hardware	No control	Full control

Table 3-4 – Software Stack and Provider/Consumer Scope of Control

The table suggests the high level functionality that a cloud platform should implement both for the consumers and the provider.

In the project context, SaaS is considered out of scope but PaaS and IaaS solutions used for implementing SCP shall implement functions for controlling and administering the resources according to the scheme reported above. This is a very high level view of the functions to implement, yet it can be used as a guideline for defining the SCP functional architecture described in the following paragraph.

3.2 Storm Clouds Platform Functional Architecture

This section describes the Storm Clouds Platform Functional Architecture. It's a functional decomposition of the system showing what functions SCP implements, how they relate to each other and the external entities (i.e. users, systems, etc.) it interacts with.
The functional architecture in not aimed to describe how SCP is implemented, therefore it doesn't mention what technical solutions will be put in place (e.g. software and hardware products, deployment strategies, etc.);

these topics will be thoroughly dealt with in a future documents. Technical information will only mentioned as examples.

The following diagram shows the Storm Clouds Platform functional architecture:

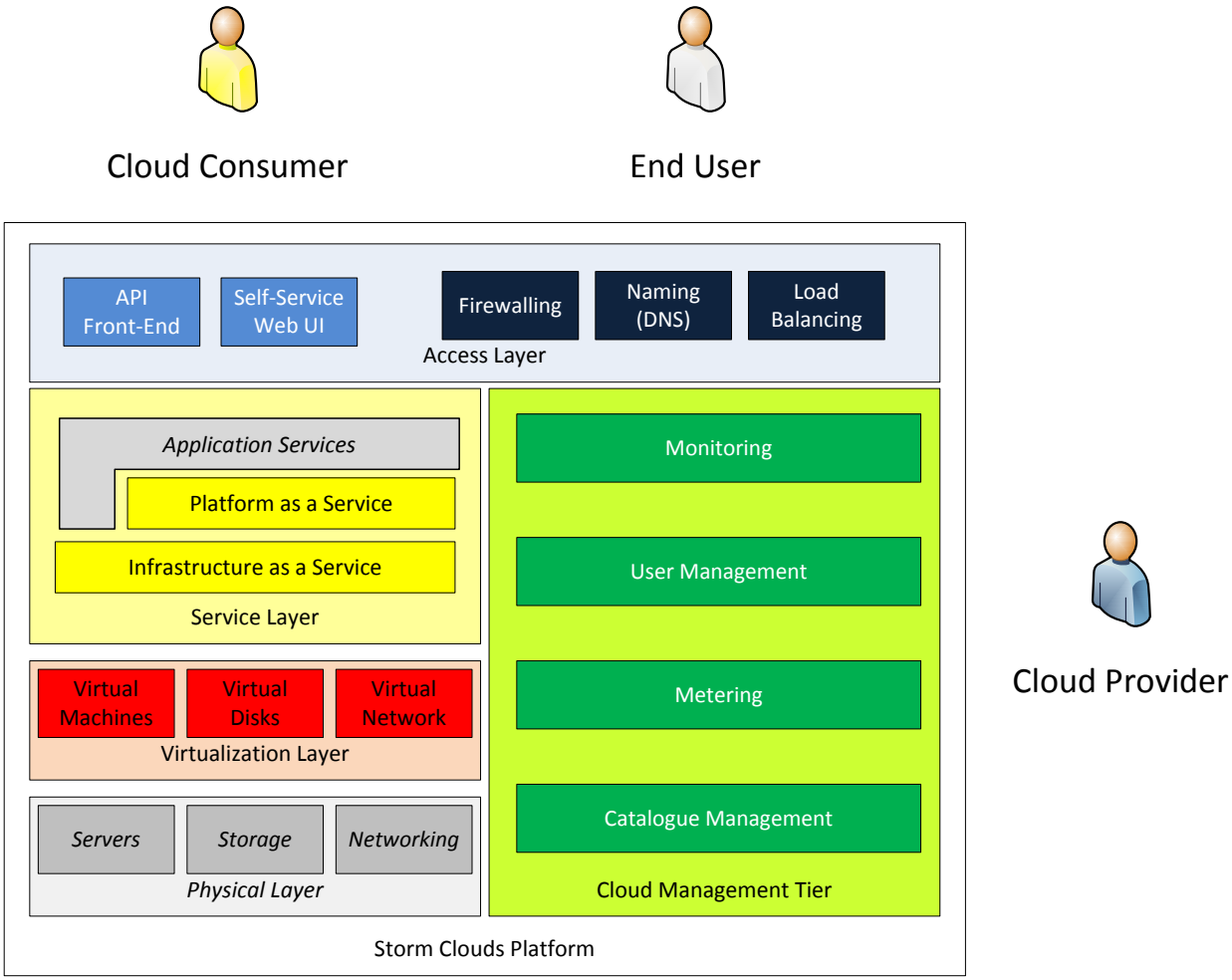

Figure 3-4 – Storm Clouds Platform Functional Architecture

The functions are implemented for users who play different roles:

- **Cloud Consumer**: the user(s) or the organization(s) that uses the services provided by SCP for implementing and deploying Application Services;
- **Cloud Provider**: the user(s) or the organization in charge of providing SCP services by implementing and maintaining the cloud;
- **End-User**: the person who uses Application Services.

It is worth noticing that **Application Services**, albeit shown in the Service Layer, are not part of the SCP architecture; they are actually hosted in the architecture, not implemented by the architecture. Application Services indicate the set of services made available to End-Users (e.g. citizens, public servants, etc.).

3.2.1 Access Layer

The Access Layer provides access to services, both cloud services and application services.

Cloud consumers manage cloud resources both via the Self Service Web User Interface (UI) and the Application Programming Interface (API) Front-End; the former is for manual operations, the latter for automatic operations implemented by cloud consumers as programs running in the cloud or outside the cloud.

Functions are made available for:

- controlling the lifecycle of the IaaS virtual objects (it includes creation and deletion of virtual machines, virtual disk storage, virtual network objects);

- uploading, downloading virtual machine images and/or application service software;
- starting/stopping virtual machines and/or application services;
- Inspecting the usage of resources.

The access layer implements functions familiar to IT departments like:

- Naming Services (DNS);
- Load Balancing;
- Firewalls.

Some of these functions may be exposed to cloud consumers and used for the deployment of Application Services.

3.2.2 Service Layer

The Service Layer actually implements the "as a Service" functions 'materializing' the computing resources for hosting the application services workload.

The Infrastructure as a Service (IaaS) component implements functions for providing cloud consumers with virtual infrastructure objects; it translates the requests coming from the Access Layer into low level operations directed to the Virtualization Layer that, in turn, is in charge of actually providing virtual objects like virtual machines, virtual disk storage and virtual network objects (e.g. Level 2 networks, virtual routers, etc.). IaaS hides the complexity of the underlying virtualization technology and implements logical high-level operations like the connection of a virtual machine to virtual networks, the attachment of a virtual disk to a virtual machine, etc. Usually these operations require a complex list of low-level commands directed to the Virtualization Layer: IaaS 'bundles' the low-level commands into single high-level commands facilitating the implementation and use of virtual infrastructure objects.

The Platform as a Service (PaaS) component implements functions for hosting and controlling specific Application Services hosted in the cloud. Generally speaking, PaaS solutions provide a framework for facilitating the development and deployment of a certain kind of applications: the SCP PaaS is meant to host web-based applications. It implements functions for up-loading and down-loading the Application Services source code, for starting and stopping applications, for inspecting the application run-time environment (e.g. log files, files, directories and databases used by the applications, etc.).

As shown in the diagram, there's no tight dependency between Application Services and PaaS. Applications can be actually deployed directly on virtual machines and can use virtual disks and virtual network objects made available by the IaaS. This arrangement is intentional because there might be applications that cannot be directly deployed on top of the PaaS.

On the other hand, the PaaS is deployed on top of the IaaS although this is not strictly required by PaaS solutions (OpenShift, CloudFoundry) that can be directly installed on bare metal servers. This option is intentional for SCP because it facilitates the management of the whole platform.

3.2.3 Virtualization Layer

The Virtualization Layer, through virtualization technologies, abstracts the physical resources and implements virtual infrastructure objects like virtual machines, virtual disks and virtual networks. This layer is usually implemented by software components installed directly on top of the operating system of the underlying physical equipment.

The primary objective of the layer is decoupling the upper layers from the physical hardware; in this manner virtual objects can be "migrated" from the physical hardware equipment where they run to another location without 'disturbing' the hosted workload.

This layer is implemented by hypervisors (for implementing virtual machines), disk virtualization technologies like iSCSI protocol based solutions (for implementing virtual disks), and network virtualization solutions, like VPN or tunnelling solutions (for implementing virtual networks).

3.2.4 Physical Layer

The physical layer is composed of the physical hardware equipment (i.e. blade servers, networking connections and disks) used for actually deploying the Storm Cloud Platform.

3.2.5 Cloud Management Tier

The cloud management tier implements functions for managing and maintaining the cloud platform and is mainly designed for the Cloud Provider.

Monitoring refers to functions for verifying the availability and the performances of the infrastructure. It is designed for verifying the working conditions of the physical resources (e.g. blade servers and network) but can also be useful for monitoring virtual resources (i.e. virtual machines) used for implementing advanced cloud services like PaaS components, virtual machines running DB engines (maybe for implementing DB as a Services components), etc.

User Management component implements functions for storing information about the Cloud Consumer(s) and for authenticating and authorizing them to access the cloud services.

Metering refers to a set of functions for acquiring measurements about the usage of cloud resources. Generally speaking, metering functions are implemented for billing purpose but in SCP context the main purpose is to gather information about the resource used for running the application service and to highlight critical situations.

Catalogue Management implements functions for storing pre-fabricated objects that cloud consumers can use for implementing application services. These can be virtual machines with some software stack installed on top, for example a LAMP (Linux, Apache, MySQL, Python) stack. Cloud consumer can implement their own application services instantiating and customizing the pre-fabricated objects.

3.3 List of Requirements

This chapter defines the requirements for the Storm Clouds Platform.

For each requirement, the following information is provided:

- **Id**: unique identifier of the requirement;
- **Description**: textual description of the requirement;
- **Type**: classification of the requirement
  - **Functional** (Fun), it can be derived from the logical model;
  - **Interface** (Int) it describes the interface with the external systems and/or users;
  - **Operational** (Ope): it specifies how the system will run and how it will communicate with the human operators;
  - **Security/Privacy** (Sec): it specifies the requirements for securing the system against threats of confidentiality, integrity and availability. e.g. intrusion detection, security patches, data encryption, multi-tenancy;
  - **Technical** (Tech): it is related to some technical e.g. languages, databases, application/web servers, tools)
- **Reference**: a reference to the 'origin' of the requirement.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [11].

Id	Description	Type	Reference
010	SCP **SHALL** be designed for providing cloud services through network connection	Fun, Int	NIST Cloud Computing Definition
020	SCP **SHOULD** adopt technical solutions that, considering the workload of the assessed applications, optimizes the usage of the physical resources used for the deployment	Tech	NIST Cloud Computing Definition
030	SCP **SHALL** allow the addition and removal of new hardware equipment without requiring full re-implementation and/or re-configuration of the already deployed infrastructure	Tech, Ope	NIST Cloud Computing Definition
040	SCP **SHALL** implement multi-tenancy and SCPP **SHALL** provide each project participants with – at least – one tenant defining the amount of resources they are allowed	Fun, Sec	NIST Cloud Computing Definition

Id	Description	Type	Reference
	to use for supporting applications		
050	SCP **SHALL** implement multi-tenancy and SCPP **SHALL** provide each project participants with – at least – one tenant defining the amount of resources they are allowed to use for supporting applications	Fun, Sec	NIST Cloud Computing Definition
060	SCCP **SHALL** provide Storm Clouds project participants with access to cloud services it implements via Internet. Access to cloud services, on the other hand, **SHALL** be regulated with some access control mechanisms based on credentials that partner's personnel are provided with.	Fun, Int, Sec	NIST Cloud Computing Definition
070	SCP **SHALL** provide any user (users and administrators) with functions for allocating and using computing resources without directly interacting with the underlying physical and/or virtualized infrastructure technology. Those functions **SHALL** be available both from a web-based interface and an API.	Fun, Int, Tech	NIST Cloud Computing Essential Characteristics
080	SCPP **SHALL** provide enough network bandwidth for supporting the applications selected for adaptation	Ope	NIST Cloud Computing Essential Characteristics (Broad Network access)
090	SCP and SCPP **SHALL** provide access to applications from various clients, using standard ad widely adopted interfaces.	Int	NIST Cloud Computing Essential Characteristics (Broad Network access)
110	SCP **SHALL** transparently use the physical resources made available by the physical infrastructure for providing the computing resources used for implementing applications.	Fun	NIST Cloud Computing Essential Characteristics (Resource Pooling)
120	SCP **SHALL** provide the ability to have some control on the location of the physical resources eventually used for deploying an application.	Fun, Sec	NIST Cloud Computing Essential Characteristics
130	SCPP **SHOULD** be located in an European Community country	Sec	NIST Cloud Computing Essential Characteristics
140	Although the cloud service provider is not provided with the ability to control the exact location of the physical resource used for implementing an application (e.g. which server or which disk), the SCP **SHALL** provide the provider with the ability of "labelling" physical resources with an abstract location (e.g. physical resources in a region, a country, etc.). It's Cloud Platform Provider's responsibility to use this feature consistently.	Tech, Ope	NIST Cloud Computing Essential Characteristics
160	SCP **SHALL** provide functions for monitoring the usage of physical resources in order to highlight overloading situations.	Ope	NIST Cloud Computing Essential Characteristics
180	SCP **SHALL** implement some basic measurement functions aimed to monitor the usage of physical and/or virtual resources (e.g. VCPUs, RAM, disk space, network traffic).	Ope	NIST Cloud Computing Essential Characteristics
190	SCP **SHALL** be designed for supporting private cloud deployment model. This is a consequence of the request of allowing project partners to implement their own cloud totally or partially reusing SCP design principles.	Ope	NIST Cloud Computing Deployment Models
200	SCPP **SHALL** be implemented as an outsourced community cloud deployment possibly leasing	Ope	NIST Cloud Computing

Id	Description	Type	Reference
	computational resources at a public cloud provider as the underlying (virtual) infrastructure for deploying SCP components and services.		Deployment Models
210	SCP **SHOULD** be implemented with technologies used for implementing public clouds available in the market.	Tech	NIST Cloud Computing Deployment Models
220	SCP **SHOULD** support hybrid cloud scenarios or – at least – permit the implementation of hybrid scenarios where workload of an application can be split between part(s) running in the cloud and part(s) running in another cloud (SCP or no SCP) or in a traditional IT infrastructure	Ope	NIST Cloud Computing Deployment Models
240	SCP **SHALL** support the portfolio with an on-line catalogue containing templates for facilitating the adaptation and deployment of cloud-based applications.	Fun	NIST Service Models
260	SCP **SHALL** implement a PaaS cloud solution for supporting web applications.	Service Delivery – PaaS	NIST Service Models
270	SCP PaaS solution **SHALL** implement security mechanisms aimed to ensure application instance separation. This means that different instances shall run on different execution environments that strictly prevent any interference (e.g. data access) among them.	Sec	NIST Service Models
280	SCP **SHALL** implement a PaaS cloud solution for hosting the highest number of candidate applications.	Fun, Tech	NIST Service Models
290	The PaaS solution provided by SCP **SHALL** be available under some sort of Open Source licensing in order to avoid vendor lock-in and allow sustainability when the project terminates.	Tech	NIST Service Models
300	The PaaS solution provided by SCP **SHOULD** be available as a public cloud solution for allowing the migration of adapted applications to a public service provider.	Ope	NIST Service Models
310	SCP **SHALL** implement a form of IaaS both for supporting those applications that cannot be integrated in the PaaS and also for implementing the PaaS solution itself.	Fun, Tech	NIST Service Models
320	The IaaS solution provided by SCP **SHALL** be available under some sort of Open Source licensing in order to avoid vendor lock-in and allow sustainability when the project terminates.	Tech	NIST Service Models
330	The IaaS solution provided by SCP **SHOULD** be available as a public cloud solution for allowing the migration of adapted applications to a public service provider.	Ope	NIST Service Models

Table 3-5 – Storm Clouds Platform Requirements

4 Storm Project Methodological Framework

This chapter briefly reports some relevant information about the Storm Clouds project framework in order to highlight aspects that have direct impact on the design of the Storm Clouds Platform (SCP) and its physical deployment, the Storm Clouds Project Platform (SCPP).

The information reported here is meant to be used for prioritizing the implementation of functions and features in order to meet the objective defined by the project phases and for giving some rough idea on the required resources – human and computational - both for the SCP design and the SCPP implementation.

Storm Clouds project duration is 36 months divided in 2 subsequent 18 month phases: **Lead Pilots Phase** and **Replication Pilots Phase.** In turn, each phase is split into two 9 months cycles.

Lead Pilots Phase is focussed on selecting and porting some candidate applications to cloud; each involved public authority (Agueda, Manchester, Thessaloniki and Valladolid) will select one or two applications per cycle originating a total number of applications ranging from 4 to 8 per cycle and from 8 to 16 during the entire phase. Services are adapted to run on top of the cloud platform, technically tested and validated with end users (i.e. citizens and public servants).

Replication Pilot Phase is aimed at consolidating a portfolio based on the applications validated in the previous phase. Applications are generalized for being reused in other cities not taking part to the project as well as scaled up to wider geographical scopes. In addition, each public authority involved in the project will chose at least one service (maybe proposed by other project participants) to be adapted and deployed. The number of total services deployed in this phase ranges from 8 to 16; they are adapted to different contexts (cities and/or geographical areas), tested and validated with end users (i.e. citizens and public servants).

When an application is modified and/or adapted, it is necessary to provide a computing environment for the development and a computing environment for testing; then, once the application is modified and/or adapted, it can be moved to the production environment and accessed by the end users. In principle, when an application is moved to production, both the development and the testing environments can be dismantled releasing the corresponding resources that can be reused for other purposes.

Analysing the project framework, we can say that the amount of resources for development and for testing never changes during the project lifespan: we need enough resources for adapting and testing up to 8 applications simultaneously. In reality the actual amount of resources in terms of RAM, number of CPUs and network traffic really depends on the applications under development/testing but, for a rough estimation, we can assume that it is a fixed amount.

On the contrary, the amount of resources for running the services in production changes according to the plan. In the first 9 month cycle no application should be in production because the participants are busy at adapting and migrating to cloud. In the second cycle, the applications adapted in the first cycle are moved to production (up to 8 applications), in the third cycle the applications adapted in the second cycle are moved to production reaching the maximum number of applications simultaneously in production, as mentioned in [3].

The following table summarizes what is described above:

Activity	Pilot Phase		Replication Phase	
	1st Cycle	2nd Cycle	3rd Cycle	4th Cycle
Development	8	8	8	0
Testing	8	8	8	0
Production	0	8	16	16

Table 4-1 – Number of Concurrent Applications in the Cloud

Another important aspect for sizing the SCPP is related to the number of users expected to use the applications. According to [3], this should be around 5,000; at this stage, it's hard to estimate the number of concurrent users and the assumption is for 2%.

5 Summary and Conclusions

This document defines the high level requirements for the Storm Clouds (Project) Platform, the computing environment providing cloud services for deploying Storm Clouds applications.

The requirements will be analysed and refined during the next project activities with the objective of designing a technical solution that fulfils the project needs.

This document and the upcoming more technical documents are to be considered as contributions to the project objective of defining "*useful guidelines on how to address the process*" [3] of transporting application services from "traditional" IT environments to cloud computing paradigm.

References

[1] "Storm Clouds - Project Web Site," [Online]. Available: http://stormclouds.eu/. [Accessed July 2014].

[2] "Storm Clouds Project - European Commission Project Page," [Online]. Available: http://ec.europa.eu/digital-agenda/en/storm-clouds-project-cloud-public-services. [Accessed April 2014].

[3] "Surfing Towards the Opportunity of Real Migration to CLOUD-based public Services," STORM CLOUDS Consortium, November 2013.

[4] "CORDIS - Free and open source software activities in European Information Society initiatives," [Online]. Available: http://cordis.europa.eu/fp7/ict/ssai/foss-home_en.html. [Accessed Apr 2014].

[5] "LAMP (software bundle) - Wikipedia Page," [Online]. Available: http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29. [Accessed Apr 2014].

[6] P. Mell and T. Granc, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, September 2011.

[7] T. G. R. P.-C. J. L.Badger, "Cloud Computing Synopsis and Recommendations," National Institute of Standards and Technology, May 2012.

[8] "Multy-tenanancy Definition - Whatis Page," [Online]. Available: http://whatis.techtarget.com/definition/multi-tenancy. [Accessed Apr 2014].

[9] "Multitenancy definition - Gartner Page," [Online]. Available: http://www.gartner.com/it-glossary/multitenancy. [Accessed Apr 2014].

[10] *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.*

[11] S. Bradner, "RFC 2119: Key words for use in RFCs to Indicate Requirement Level," 1997. [Online]. Available: https://www.ietf.org/rfc/rfc2119.txt. [Accessed Apr 2014].

