D1.1

Requirements Analysis Report
Version 1.1 - 24/02/2011

WP 1 Cloud4SOA Reference Framework
Dissemination Level: Public

Lead Editor: Nikos Loutas, CERTH

Status: Final

Seventh Framework Programme of the European Community

Proposal/Contract no.: 257953
Context

<table>
<thead>
<tr>
<th>WP 1</th>
<th>Cloud4SOA Reference Framework</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Task 1.1</th>
<th>State-of-the-Art and Requirements Analysis</th>
</tr>
</thead>
</table>

| Dependencies | D1.1 documents the review of the State-of-the-Art and the prioritized list of requirements that will be addressed by the Cloud4SOA reference framework. |

Contributors: Nikos Loutas, Eleni Kamateri, Maria Zotou, Dimitris Zeginis, Konstantinos Tarabanis (CERTH); S. Bocconi, Esen Mogulkoc (Cyntelix);
Reviewers: Paul Polydoras (SingularLogic), Philipp Strube (cloudControl)
Approved by: QM

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Authors</th>
<th>Sections Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>28/09/10</td>
<td>N. Loutas, K. Tarabanis</td>
<td>ToC</td>
</tr>
<tr>
<td>0.2</td>
<td>25/10/10</td>
<td>E. Kamateri, N. Loutas</td>
<td>D. Sections 3,4,6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zeginis, M. Zotou</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>28/10/10</td>
<td>S. Bocconi, M. Zotou</td>
<td>ToC of section 5</td>
</tr>
<tr>
<td>0.4</td>
<td>11/11/10</td>
<td>E. Kamateri, N. Loutas</td>
<td>D. Sections 3,4,6 updated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zeginis, M. Zotou</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>30/11/10</td>
<td>E. Kamateri</td>
<td>Sections 3,4,6 updated</td>
</tr>
<tr>
<td>0.6</td>
<td></td>
<td>E. Kamateri, M. Zotou</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>24/01/11</td>
<td>N. Loutas, K. Tarabanis</td>
<td>Proof-reading and editing of D1.1</td>
</tr>
<tr>
<td>0.8</td>
<td>27/01/11</td>
<td>E. Kamateri</td>
<td>Section 4.5 – Cyntelix contribution included.</td>
</tr>
<tr>
<td>0.9</td>
<td>31/01/11</td>
<td>N. Loutas, E. Kamateri,</td>
<td>Sections 3,4,6 updated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M. Zotou</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>08/02/11</td>
<td>N. Loutas</td>
<td>Final draft sent to the reviewers</td>
</tr>
<tr>
<td>1.1</td>
<td>21/2/11</td>
<td>N. Loutas, E. Kamateri,</td>
<td>Reviewers’ recommendations addressed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M. Zotou, D. Zeginis,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>K. Tarabanis</td>
<td></td>
</tr>
</tbody>
</table>
# Table of Contents

1. Introduction ........................................................................................................................................... 16  
   1.1. Document Scope ........................................................................................................................... 16  
   1.2. Methodology ................................................................................................................................. 16  
   1.3. Overview ...................................................................................................................................... 18  

2. State of the Art in Cloud Computing Interoperability .................................................................... 19  
   2.1. Cloud Computing Interoperability ............................................................................................ 19  
       2.1.1. Definitions for Cloud Computing Interoperability ............................................................. 20  
       2.1.2. Types of Cloud Computing Interoperability ....................................................................... 22  
       2.1.3. Standards for Cloud Computing Interoperability ................................................................. 23  
   2.2. Cloud PaaS Interoperability ......................................................................................................... 27  
       2.2.1. Platform as a Service in a nutshell ....................................................................................... 27  
       2.2.2. Cloud PaaS Interoperability Problems .............................................................................. 29  
   2.3. Interoperable Cloud Computing Architectures ......................................................................... 30  
       2.3.1. Industry-driven Approaches ............................................................................................... 30  
           2.3.1.1. eCloudManager ........................................................................................................... 31  
           2.3.1.2. Aneka ............................................................................................................................ 32  
           2.3.1.3. Cloud Exchange Federated Cloud .............................................................................. 33  
           2.3.1.4. SaaS-PF ......................................................................................................................... 34  
           2.3.1.5. Open PaaS ................................................................................................................... 35  
           2.3.1.6. Red Hat Reference Cloud Computing Architecture .................................................. 35  
           2.3.1.7. Cisco Reference Cloud Computing Architecture ....................................................... 36  
           2.3.1.8. IBM Reference Cloud Computing Architecture ......................................................... 37  
           2.3.1.9. Oracle Architecture for private PaaS ................................................................... 38  
       2.3.2. Academia-driven Approaches ............................................................................................... 38  
           2.3.2.1. The Cloud Development Stack model ......................................................................... 38  
           2.3.2.2. Chappell’s three-partite model ..................................................................................... 39  
           2.3.2.3. The next generation Cloud architecture ....................................................................... 40  
           2.3.2.4. Charlton’s Cloud computing reference architecture .................................................... 41  
           2.3.2.5. Cloud computing reference model ............................................................................... 42  
           2.3.2.6. Sambyal et al. Cloud Computing model ...................................................................... 43  
           2.3.2.7. Adaptive PaaS Architecture .......................................................................................... 44  
           2.3.2.8. Cloud deployment model ............................................................................................... 44  
       2.3.3. R&D project-driven Approaches ............................................................................................ 45  
           2.3.3.1. 4CaaSt ............................................................................................................................ 45
2.3.3.2. CumuloNimbo ............................................................................................................ 46
2.3.3.3. Cloud-TM .................................................................................................................... 47
2.3.3.4. mOSAIC ...................................................................................................................... 47
2.3.3.5. CONTRAIL ................................................................................................................ 48
2.3.3.6. Vision Cloud ............................................................................................................... 49
2.3.3.7. REMICS ................................................................................................................... 50
2.3.3.8. RESERVOIR .............................................................................................................. 51
2.3.3.9. SLA@SOI ................................................................................................................... 52
2.3.3.10. SITIO ....................................................................................................................... 53
2.3.3.11. NEXOF ................................................................................................................... 54
2.3.3.12. Cloud@Home ........................................................................................................ 55
2.3.3.13. SOA4All .................................................................................................................. 57
2.3.4. Discussion ............................................................................................................................. 58
2.4. Semantic Models for Cloud Computing ............................................................................... 60
   2.4.1. Cloud Service Models .................................................................................................. 60
   2.4.1.1. Cloud Computing Services Taxonomy ................................................................... 60
   2.4.1.2. Cloud Computing Services Ontology .................................................................... 60
   2.4.1.3. Intel Cloud Computing Services Taxonomy .......................................................... 61
   2.4.1.4. Rimal et al. Cloud Computing Services Taxonomy ............................................. 61
   2.4.1.5. Cloud Computing stack .......................................................................................... 62
   2.4.2. Cloud Resource Models .............................................................................................. 63
   2.4.2.1. Han & Sim Cloud resource model ........................................................................... 63
   2.4.2.2. Haase et al. Cloud resource model ......................................................................... 64
   2.4.2.3. Oracle Cloud resource model .................................................................................. 65
   2.4.2.4. DeltaCloud resource model .................................................................................... 65
   2.4.2.5. OCCI Cloud resource model ................................................................................... 66
   2.4.2.6. Amazon Cloud resource model ............................................................................... 66
   2.4.3. APIs for Cloud Computing ............................................................................................ 67
   2.4.3.1. IaaS APIs .................................................................................................................. 67
   2.4.3.2. PaaS APIs .................................................................................................................. 71
   2.4.3.3. Broker APIs .............................................................................................................. 76
   2.4.3.4. Cloud-Standard Cloud API ................................................................................... 78
   2.4.3.5. Elastic Modeling Languages ..................................................................................... 79
   2.4.4. Cloud Computing Meta-models ................................................................................... 80
   2.4.4.1. Meta-model of Cloud computing components and resources ................................ 80
   2.4.4.2. Simplified Cloud meta-model .................................................................................. 81
List of Figures

Figure 1: The Cloud4SOA Requirements Elicitation ................................................................. 16
Figure 2: Portability and automation across Cloud layers [3] .................................................... 22
Figure 3: eCloudManager Architecture [40] ............................................................................ 32
Figure 4: Overview of the Aneka’s Framework [41] ................................................................ 33
Figure 5: Federated network of Clouds mediated by a Cloud Exchange [42] ............................ 34
Figure 6: SaaS-PF Architecture [43] .......................................................................................... 34
Figure 7: The design of an open PaaS system [35] ................................................................. 35
Figure 8: A reference architecture released by Red Hat [44] .................................................... 36
Figure 9: Cisco’s Cloud Reference Architecture [45] ............................................................... 36
Figure 10: IBM’s reference architecture [46] ............................................................................ 37
Figure 11: A reference architecture for PaaS private Clouds released by Oracle [47] .............. 38
Figure 12: The Cloud Development Stack model [48] ............................................................. 39
Figure 13: A modern application platform [49] ....................................................................... 40
Figure 14: The next generation Cloud architecture [50] .......................................................... 40
Figure 15: Cloud Reference Architecture [51] ....................................................................... 41
Figure 16: Cloud Computing Reference Model [10] ............................................................... 42
Figure 17: The Cloud Platform tier as part of the Enablement Model [10] ............................... 43
Figure 18: A new Cloud computing model [16] .................................................................... 43
Figure 19: An adaptive PaaS architecture [26] ....................................................................... 44
Figure 20: Application and Deployment descriptor [52] ......................................................... 44
Figure 21: Architecture proposed by the 4CaaSt project [53] .................................................... 46
Figure 22: Architecture of the CumuloNimbo Project [53] ....................................................... 46
Figure 23: Architectural view of the Cloud-TM platform [53] .................................................... 47
Figure 24: Platform architecture of the mOSAIC project [53] ................................................... 48
Figure 25: Integrating multiple independent Clouds into a Federated Cloud [53] ................... 49
Figure 26: The VISION Cloud infrastructure [53] ................................................................. 50
Figure 27: Overview of the REMICS project’ research approach [53] ...................................... 50
Figure 28: Architecture proposed by the RESERVOIR project [57] ....................................... 51
Figure 29: The interaction of SLA stakeholders proposed by SLA@SOI [58] ............................ 53
Figure 30: Architecture proposed by SITIO [59] .................................................................... 54
Figure 31: NEXOF Reference Architecture [60] .................................................................... 55
Figure 32: Basic architecture of the Cloud@Home project [61] .............................................. 56
Figure 33: Configuration of the Cloud@Home system [61] .................................................... 56
Figure 34: The SOA4All Architecture [62] ............................................................................ 57
Figure 35: Cloud Ontology [63] .......................................................... 61
Figure 36: Cloud Computing Taxonomy by Intel [64] ......................................................... 61
Figure 37: A Cloud Computing Taxonomy [65]......................................................................... 62
Figure 38: The Cloud Computing stack [66] ........................................................................... 63
Figure 39: Cloud Resource Model [71] .................................................................................... 64
Figure 40: Cloud Resource Model [40] .................................................................................... 64
Figure 41: Oracle Cloud Resource Model [67] ........................................................................ 65
Figure 42: DeltaCloud Resource Model [68] ........................................................................... 66
Figure 43: OCCI Resource Model [69] ..................................................................................... 66
Figure 44: Amazon Resource Model ......................................................................................... 67
Figure 45: The Amazon API ..................................................................................................... 68
Figure 46: Rackspace API ......................................................................................................... 68
Figure 47: The GoGrid API ........................................................................................................ 69
Figure 48: The vCloud API ....................................................................................................... 69
Figure 49: The Oracle Cloud Resource Model API ................................................................. 70
Figure 50: The ElasticHost API ............................................................................................... 70
Figure 51: The FlexiScale API ................................................................................................. 71
Figure 52: The Amazon PaaS API ........................................................................................... 72
Figure 53: The Azure API ........................................................................................................ 72
Figure 54: The Google App Engine API ................................................................................... 73
Figure 55: The Heroku API ...................................................................................................... 73
Figure 56: The Oracle PaaS API ............................................................................................. 74
Figure 57: The VMWare PaaS ................................................................................................. 74
Figure 58: The SalesForce API3 .............................................................................................. 75
Figure 59: The Red Hat API ..................................................................................................... 75
Figure 60: The DeltaCloud API [85] ....................................................................................... 76
Figure 61: The RightScale API [86] ......................................................................................... 77
Figure 62: The Enomaly API [87] ............................................................................................ 77
Figure 63: The OpenNebula API [89] ...................................................................................... 78
Figure 64: Cloud-Standard Cloud API [90] ............................................................................. 79
Figure 65: The three current Elastic Modeling Languages [51] .............................................. 79
Figure 66: Meta-model of Cloud computing components and resources [91] ..................... 80
Figure 67: Simplified Cloud meta-model [92] .......................................................................... 81
Figure 68: Nova Spivak’s timeline of the past, present and future of the Web [118] .......... 90
Figure 69: FAST Editor to create widget Mash-ups ................................................................. 92
Figure 70: The SOA4All process editor (screenshot) .............................................................. 93
**List of Tables**

Table 1: Interoperability – Standardization Bodies Initiatives ................................................................. 27
Table 2: Types of PaaS solutions [26] ........................................................................................................... 29
Table 3: Related projects *vis a vis* Cloud4SOA ....................................................................................... 58
Table 4: Cloud Service Models .................................................................................................................... 81
Table 5: Cloud Resource Models ................................................................................................................ 83
Table 6: Cloud IaaS APIs ............................................................................................................................ 83
Table 7: Cloud PaaS APIs ........................................................................................................................... 84
Table 8 Cloud Brokers APIs ....................................................................................................................... 84
Table 9: Core requirements of any Cloud system [131] ........................................................................ 98
Table 10: Requirements derived from lifecycle usage scenarios [131] ................................................ 98
Table 11: Stakeholders per architecture .................................................................................................... 102
Table 12: Functional Requirements collected from the literature ........................................................ 103
Table 13: Non-Functional Requirements collected from the literature .................................................. 105
Table 14: Functional requirements collected from the US ................................................................ 110
Table 15: Non-functional requirements collected from the US ........................................................... 111
Table 16: Integrated list of Functional Requirements ........................................................................... 113
Table 17: Integrated list of Non-Functional Requirements ...................................................................... 115
Table 18: Number of functional and non-functional requirements per priority level .................. 119
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4CaaSt</td>
<td>Building the PaaS Cloud of the Future</td>
</tr>
<tr>
<td>4GL</td>
<td>Fourth-generation programming language</td>
</tr>
<tr>
<td>AM</td>
<td>Autonomic Manager</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>apps</td>
<td>applications</td>
</tr>
<tr>
<td>ASP</td>
<td>Application Service Provider</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
</tr>
<tr>
<td>BPM</td>
<td>Business Process Management</td>
</tr>
<tr>
<td>CaaS</td>
<td>Computing as a Service</td>
</tr>
<tr>
<td>CCE</td>
<td>Cloud Computing Environment</td>
</tr>
<tr>
<td>CCIF</td>
<td>Cloud Computing Interoperability Forum</td>
</tr>
<tr>
<td>CC-RM</td>
<td>Cloud Computing Reference Model</td>
</tr>
<tr>
<td>CCWG</td>
<td>Cloud Computing Working Group</td>
</tr>
<tr>
<td>CIF</td>
<td>Cloud Industry Forum</td>
</tr>
<tr>
<td>Cloud@Home</td>
<td>Cloud Computing over Unreliable, Shared Resources</td>
</tr>
<tr>
<td>Cloud4SOA</td>
<td>A Cloud Interoperability Framework and Platform for user-centric, semantically-enhanced service-oriented applications design, deployment and distributed execution</td>
</tr>
<tr>
<td>Cloud-TM</td>
<td>Cloud-TM: A novel programming paradigm for Cloud computing</td>
</tr>
<tr>
<td>CMWG</td>
<td>Cloud Management Working Group</td>
</tr>
<tr>
<td>CONTRAIL</td>
<td>Open Computing Infrastructures for Elastic Services</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>CTT</td>
<td>Concur Task Tree</td>
</tr>
<tr>
<td>CumuloNimbo</td>
<td>A Highly Scalable Transactional Multi-Tier Platform as a Service</td>
</tr>
<tr>
<td>DaaS</td>
<td>Data as a Service</td>
</tr>
<tr>
<td>DCI</td>
<td>Distributed Computing Infrastructure</td>
</tr>
<tr>
<td>DG</td>
<td>Desktop Grid</td>
</tr>
<tr>
<td>DHTML</td>
<td>Dynamic HTML</td>
</tr>
</tbody>
</table>
DMTF  Distributed Management Task Force
DSS   Distributed Storage System
DSTM  Distributed Software Transactional Memory
EC    European Commission
ECP   Elastic Computing Platform
ECML  Elastic Computing Modeling Language
EDGI  European Desktop Grid Initiative
EDML  Elastic Deployment Modeling Language
EGI   European Grid Initiative
EGI-InSPIRE  European Grid Initiative - Integrated Sustainable Pan-European Infrastructure for Researchers in Europe
EMI   European Middleware Initiative
EMML  Elastic Management Modeling Language
ENISA European Network and Information Security Agency
cTOM  enhanced Telecom Operations Map
ETSI  European Telecommunications Standards Institute
EU    European Union
EUD   End-user Development
EzWeb Put a Face on Services
FAST  Fast and Advanced Storyboard Tools
FCAPS Fault, Configuration, Accounting, Performance and Security
FR    Functional Requirements
GICTF Global Inter-Cloud Technology Forum
HaaS  Hardware as a Service
HPC   High Performance Computing
HTA   Hierarchical Task Analysis
HTML  Hyper Text Markup Language
HTTP  Hypertext Transfer Protocol
IaaS  Infrastructure as a Service
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>ID</td>
<td>Identification Document</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IGE</td>
<td>Initiative for Globus in Europe</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization.</td>
</tr>
<tr>
<td>IT</td>
<td>Internet Technology</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Platform, Enterprise Edition</td>
</tr>
<tr>
<td>JMX</td>
<td>Java Management Extensions</td>
</tr>
<tr>
<td>mOSAIC</td>
<td>Open-Source API and Platform for Multiple Clouds</td>
</tr>
<tr>
<td>NCOIC</td>
<td>Network Centric Operations Industry Consortium</td>
</tr>
<tr>
<td>NEXOF-RA</td>
<td>NESSI Open Framework – Reference Architecture</td>
</tr>
<tr>
<td>NFR</td>
<td>Non-Functional Requirements</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organization for the Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>OCC</td>
<td>Open Cloud Consortium</td>
</tr>
<tr>
<td>OCCI</td>
<td>Open Cloud Computing Interface</td>
</tr>
<tr>
<td>OGF</td>
<td>Open Grid Forum</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OVF</td>
<td>Open Virtualization Format</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PAL</td>
<td>Platform Abstraction Layer</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PHP</td>
<td>Hypertext Preprocessor</td>
</tr>
<tr>
<td>POXO</td>
<td>Plain Old Java Object</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>REMICS</td>
<td>REuse and Migration of legacy applications to Interoperable Cloud Services</td>
</tr>
</tbody>
</table>
RESERVOIR Resources and Services Virtualization without Barriers

REST Representational State Transfer

RSS Really Simple Syndication

SaaS Software as a Service

SAN Storage Area Network

SCOPE Systems, Capabilities, Operations, Programs and Enterprise

SDK Software Development Kits

SID Shared Information/Data Model

SIENA Standards and Interoperability for eInfrastructure implementation initiative

SITIO Semantic Business Processes based on Software-as-a-Service and Cloud Computing

SLA@SOI Empowering the service industry with SLA-aware infrastructures

SLAs Service Level Agreements

SMEs Small and Medium Enterprises

SOA Service Oriented Architecture

SOA4All Service Oriented Architectures for All

SOAP Simple Object Access Protocol

SQL Structured Query Language

SSO Single Sign-On

StratusLab Enhancing Grid Infrastructures with Virtualization and Cloud Technologies

TAM Telecom Application Map


UI User Interface

UML Unified Modeling Language

URI Universal Resource Identifier

US Usage Scenario

VDC Virtual Data Centers

VEEH Virtual Execution Environment Host

VEEM Virtual Execution Environment Manager
VENUS-C     Virtual multidisciplinary EnviroNments USing Cloud Infrastructures
Vision Cloud Virtualized Storage Services Foundation for the Future Internet
VM         Virtual Machine
VNet       Virtual Networks
VPC        Virtual Private Clouds
W3C        World Wide Web Consortium
WP         Work Package
XHTML      eXtensible HyperText Markup Language
XML        Extensible Markup Language
XtreemOS   Building and Promoting a Linux-based Operating System to Support Virtual Organizations for Next Generation Grids
Executive Summary

The present document is Deliverable 1.1 “Requirements Analysis Report” (henceforth referred to as D1.1) of the Cloud4SOA project. The main objective of this document is to gather the requirements that will be addressed by the Cloud4SOA Reference Framework thus contributing to the realization of the Cloud4SOA vision.

The vision of Cloud4SOA is to open up the Cloud market to small-medium European PaaS providers and strengthen their market position and to treat the vendor-lock in problem. Cloud4SOA will thus enhance Cloud-based application development, deployment and migration by semantically interconnecting heterogeneous Platform as a Service (PaaS) offerings both within the same as well as across different Cloud PaaS providers and will facilitate the access of Cloud-based application developers to the PaaS offering that best matches their computational needs.

The Cloud4SOA consortium

In order to collect the requirements that will drive the development of the Cloud4SOA Reference Framework, the following steps have been carried out: (i) Collection and structured review of the relevant State of the Art; (ii) Development of two usage scenarios; (iii) Elicitation of requirements from (i) and (ii); and (iv) Prioritization of the requirements using scorecards.

The first step was to carry out an extended review of the State of the Art in the following research fields covered by Cloud4SOA in order to identify gaps, deficiencies, needs and problems, and thus elicit requirements: Cloud computing problems, focusing mostly on semantic interoperability issues at the PaaS-PaaS level; Semantic service, resource and data models to support the semantic annotation of Cloud services and resources; and Intelligent and adaptive service front-ends for the development of service-oriented Cloud-based applications. The outcome of step i is documented in section 2.

In order to actively involve the consortium Cloud stakeholders in the requirements elicitation process two usage scenarios have been developed. They focus on the functionalities that the Cloud4SOA platform should offer to address specific requirements of the trial partners, i.e. FIT, PTIN and RomTelecom, and cloudControl, as well as on the expected benefits of the Cloud4SOA solution. The outcome of step ii is documented in section 3.1.

Once these two parallel steps were completed, two sets each one containing a list of functional and a list of non-functional requirements were identified (see sections 2.6 and 3.2) and merged into a single list (see section 4).

Finally, a prioritization exercise was carried out. Three priority levels were identified:

- Top priority was given to all the requirements that originated both in the usage scenarios and the literature review.
- Medium priority was assigned to all the requirements coming only from the usage scenarios.
- Low priority was given to all the requirements that derived only from the literature review.

The top and medium priority requirements are the ones that will be addressed first in the context of the Cloud4SOA Cloud Semantic Interoperability Framework (D1.2) and the Cloud4SOA Reference Architecture (D1.3).
1. **Introduction**

1.1. **Document Scope**

The present document is Deliverable 1.1 “Requirements Analysis Report” (henceforth referred to as D1.1) of the Cloud4SOA project. The main objective of this document is to gather the requirements that will be addressed by the Cloud4SOA Reference Framework thus contributing to the realization of the Cloud4SOA vision.

The vision of Cloud4SOA is to open up the Cloud market to small-medium European PaaS providers and strengthen their market position and to treat the vendor-lock in problem. Cloud4SOA will thus enhance Cloud-based application development, deployment and migration by semantically interconnecting heterogeneous Platform as a Service (PaaS) offerings both within the same as well as across different Cloud PaaS providers and will facilitate the access of Cloud-based application developers to the PaaS offering that best matches their computational needs.

The Cloud4SOA consortium

1.2. **Methodology**

In order to collect the requirements that will drive the development of the Cloud4SOA Reference Framework, the following steps have been carried out (Figure 1):

i. Collection and structured review of the relevant State of the Art;

ii. Development of two usage scenarios;

iii. Elicitation of requirements from (i) and (ii); and

iv. Prioritization of the requirements.

![Figure 1: The Cloud4SOA Requirements Elicitation](image)

As explained above, the first step was to carry out an extended review of the State of the Art in the following research fields covered by Cloud4SOA in order to identify gaps, deficiencies, needs and problems, and thus elicit requirements:

- Cloud computing architectures, focusing mostly on semantic interoperability issues at the PaaS and IaaS levels;

- Semantic service, resource and data models to support the semantic annotation of Cloud services and resources; and

- Intelligent and adaptive service front-ends for the development of service-oriented Cloud-based applications and the governance and monitoring of semantically interoperable Cloud platforms.
We started by searching the major research databases of computer science, i.e. ACM Digital Library, IEEE Xplore, SpringerLink, ScienceDirect and Google Scholar using keywords such as Cloud computing, PaaS, interoperability, architectures, semantic, abstraction, interoperation, intelligent, adaptive etc. We preferred publications dated from 2007 (as according to Google Trends’ search and news reference volume data the term ‘Cloud computing’ started becoming popular in 2007 [1]) till December 2010. The references of the selected papers were checked and additional papers were found. Electronic articles written in blogs and electronic journals such as Cloud Computing Journal\(^1\), Cloud Security\(^2\) and Network World\(^3\) referring to Cloud computing interoperability were also reviewed. Finally, initiatives coming from standardization bodies, leading vendors and funded projects were also included in the survey.

This resulted in a collection of more than 200 publications that included (a) conference, workshop and symposium papers, (b) journal articles, (c) electronic articles and (d) technical reports and white papers. Around 120 publications were finally selected as the most relevant. Furthermore, a total of 48 sites were selected, which discuss initiatives from standardization bodies, consortia, researching organizations/institutes and funded projects. The outcome of step \(i\) is documented in section 2.

In order to actively involve the consortium Cloud stakeholders, e.g. the Cloud4SOA trial partners, in the requirements elicitation process two usage scenarios have been developed. They focus on the functionalities that the Cloud4SOA platform should offer to address specific requirements of the trial partners, i.e. FIT, PTIN and RomTelecom, and cloudControl, as well as on the expected benefits of the Cloud4SOA solution. The outcome of step \(ii\) is documented in section 3.1.

Once these two parallel steps were completed, two sets each one containing a list of functional and a list of non-functional requirements (following the ISO-9126 standard [2]) were identified (see sections 2.6 and 3.2).

Finally, a prioritization exercise was carried out. First the two lists of functional requirements were merged into a single list. Likewise, the non-functional requirements were merged into one single list of non-functional requirements. Three priority levels were identified:

- Top priority was given to all the requirements that originated both in the usage scenarios and the literature review.
- Medium priority was assigned to all the requirements coming only from the usage scenarios.
- Low priority was given to all the requirements that derived only from the literature review.

The prioritized lists of functional and non-functional requirements can be found in section 4.

The top and medium priority requirements are the ones that will be addressed first in the context of the Cloud4SOA Cloud Semantic Interoperability Framework (D1.2) and the Cloud4SOA Reference Architecture (D1.3).

---

\(^1\) http://Cloudcomputing.sys-con.com/
\(^2\) http://cloudsecurity.trendmicro.com/
\(^3\) http://www.networkworld.com/
1.3. Overview

The rest of the document is divided into four sections.

Section 2 documents the current State of the Art in architectures and models to address Cloud computing interoperability and extracts requirements that will be considered by the Cloud4SOA consortium. To this end, we initially investigate how Cloud computing interoperability has emerged as a high priority for Cloud technology, including the rationale, definitions of “Cloud computing interoperability” and standardization efforts. Moreover, we introduce the main Cloud computing interoperability issues arising on the PaaS layer of the Cloud stack focusing on semantic aspects. Next, we review Cloud computing architectures that aim to address interoperability issues. Furthermore, the existing Cloud service models and resource models, APIs as well as meta-models used for Cloud computing are presented. A review of intelligent and adaptive interfaces for Cloud computing follows. Section 2 concludes with a list of functional and non-functional requirements.

Section 3 presents Cloud4SOA’s main stakeholders and usage scenarios. Functional and non-functional requirements derive from the usage scenarios.

Section 4 merges the requirements collected in sections 2 and 3 in an all-inclusive list and discusses their prioritization.

Finally, Section 5 discusses our findings and concludes the requirements analysis.
2. **State of the Art in Cloud Computing Interoperability**

This section reports the current State of the Art in Cloud computing interoperability focusing on semantic aspects. Section 2.1 discusses the rationale behind Cloud computing interoperability and how it has emerged as a high priority for Cloud technology. Moreover, it presents several definitions for Cloud computing interoperability as well as several standardization initiatives in the field.

In section 2.2, Cloud computing interoperability issues arising in PaaS layer of the Cloud stack are investigated, exploring also the current state in PaaS layer.

In section 2.3, Cloud computing architectures aiming to address Cloud (semantic) interoperability are presented.

Section 2.4 studies the existing Cloud service models, resource models and APIs that are used by Cloud implementations as well as meta-models and concepts (including relationships and attributes) used for the modelling of Cloud platforms.

Section 2.5 documents the State of the Art in intelligent and adaptive interfaces. Furthermore, it discusses the issues that need to be addressed and the gaps that need to be filled (missing functionality detected in the previous section) for the successful realization of our vision.

Lastly, section 2.6 reviews the basic functionality of an (ideal) interoperable PaaS system. The State of the Art analysis will then feed the requirements elicitation process extracting Cloud4SOA’s main functionalities for the development of semantically interconnected heterogeneous PaaS offerings.

### 2.1. Cloud Computing Interoperability

The current Cloud computing solutions have not been built with interoperability as a primary concern [3]. Current Cloud computing offerings usually “lock” customers into a single Cloud infrastructure, platform or application, preventing the portability of data or software created by them. Even if portability is supported, it is rarely used by customers due to its complexity and high switching costs. The European Network and Information Security Agency (ENISA) has also recognized the lock-in problem as a high risk that Cloud infrastructures entail, [4] as it will be later on analyzed.

The increasing competition between the leading vendors in the Cloud market, such as Amazon, Microsoft, Google and SalesForce, each of which promotes its own, incompatible Cloud standards and formats [5], prevents them from agreeing on a widely accepted, standardized way to input/output Cloud details and specifications. For the same reason Small and Medium Enterprises (SMEs) are reluctant to enter the Cloud market.

However, an interoperable Cloud environment would benefit customers, as they could migrate their data and applications between Cloud providers without setting data at risk. Moreover, they would be able to compare Cloud offerings with different characteristics, such as resource, pricing or Quality of Server (QoS) model, and to choose the most cost-effective offering. Besides creating a competitive market for Cloud customers, interoperability could also attract more SMEs to the Cloud market and new business models among Cloud providers could emerge according to demand. For example, an unexpected increase in processing power capacity could force Cloud providers to cooperate in order to overcome the problem of limited resources. Otherwise, SMEs...
would seem unreliable to provide the negotiated QoS, leading consumers to rely on big players for hosting their services and data [6].

Besides enterprises and customers, interoperable Cloud environments would benefit academia, as data and application sharing would be facilitated.

According to existing literature, overcoming the interoperability problem would probably result in the creation of the InterCloud, the Cloud of Clouds [7-9], which refers to globally interconnected cooperating Clouds.

The establishment of an open “Cloud vision” requires the development of standards focused on software portability and interoperability between Cloud providers. In order to achieve this, standardization bodies should properly cooperate with academia, governments and industrial sector on advancing Cloud computing interoperability standards. The areas of Cloud computing that should be involved range from Application Programming Interfaces (APIs) to governance and run-time standards [10]. Furthermore, Cloud services should be accessed via marketplaces where customers can view, compare and select the preferable Cloud services. Therefore, additional standards, Cloud performance benchmarks and standardized pricing models should be adopted. The performance benchmarks will guide consumers on how Cloud computing can increase asset utilization, resource optimization and other performance indicators. Cloud service discovery, portability, onboarding and offboarding models and Cloud provider abstraction will also enable seamless switching of Cloud providers where onboarding and offboarding models describe how applications, data, business process, and business operations can be moved into a Cloud deployment [10].

The “Grids, Clouds and Service Infrastructures” workshop, set up by the Cloud computing working group of the European Telecommunications Standards Institute (ETSI TC CLOUD4), has focused on exploring Cloud computing platforms’ open issues and on developing standards. One of the main outcomes identified the main Cloud computing sub-areas where the need for standardization is high, including portability, interoperability and APIs, support for building, modelling, testing and deploying applications, clearly defined Service Level Agreements (SLAs) and fit for business use [11].

The rest of this section presents and analyses several definitions and categorizations of Cloud computing interoperability and investigates further the basic aspects of interoperability. Moreover, initiatives derived from Cloud computing standardization bodies focusing on interoperability are reviewed.

2.1.1. Definitions for Cloud Computing Interoperability

According to the European Commission (EC), interoperability is defined as the ability of Information and Communication Technology (ICT) systems and of the business processes they support to exchange data and to enable the distribution of information and knowledge [12].

In Cloud computing, the interoperability between two different Cloud providers refers to their ability to cooperate or else interoperate [13], thus establishing a federation of Clouds. Therefore, interoperability is a prerequisite for cooperation. In the reviewed literature there are several attempts to scope, address and define Cloud computing interoperability.

The Use Case Cloud Computing Discussion Group emphasizes that interoperability requires the conveyed information to be understood by the receiving systems. In the case of Cloud

http://www.etsi.org/WebSite/Technologies/GRID_CLOUD.aspx

© Cloud4SoA consortium

page 20 of 131
computing platforms, this means that “interoperability is the ability to write code that works with more than one Cloud provider simultaneously, regardless of the differences between the providers” meaning that there is the need of an API in order for the code to interact with any kind of system.[14].

While the previous definition comes short to explain what “code” is, Foley provides a detailed description of “code” and the need to be integrated for the successful interoperation of different Cloud providers. Specifically, he states that Cloud computing interoperability is “the ability to move data, applications and virtual servers from one Cloud computing environment to another”[15].

Most of the Cloud providers, especially the big players in the Cloud market, use different management APIs to differentiate their products and prevent customers from moving to another Cloud provider. Therefore, this aspect should be included in a definition for Cloud computing interoperability, i.e. “interoperability refers to customers’ ability to use the same artifacts, such as management tools, virtual server images, and so on, with a variety of Cloud computing providers and platforms”[16].

A similar definition is given by Cerf, who states that “Cloud computing interoperability has a number of dimensions including communication interoperability (HTTP, SOAP, REST), Cloud management and interaction API interoperability (createImage, terminateImage, etc.), and image portability”[17].

Another approach investigating interoperability at the management level of Cloud computing platforms is presented in [18]. In this case, interoperability is defined as “modularity and flexibility to interface easily with any service or technology in the virtualization and Cloud ecosystem” and “standardization to avoid vendor lock-in and to create a healthy community”.

However, Cloud computing interoperability includes more characteristics than these reported above. For example, Goyal gives the following definition, identifying a number of basic concepts: “Cloud computing interoperability includes the ability of some application code to run on more than one provider CCE, regardless of the differences between the CCE. It also includes process execution, security, portability, migration/cloning control, standards, transparency, and manageability and regulatory compliance”[19].

The ETSI TC CLOUD defines interoperability as the cooperation of multiple Clouds to support an application: “Interoperability is closely related to portability… Interoperability involves software and data simultaneously active in more than one Cloud infrastructure, interacting to serve a common purpose”[20].

According to the literature review, Cloud computing interoperability, compatibility and portability are closely related terms and often confused [20]. Therefore, Cohen clarifies the similarities and the differences among these terms in an attempt to exemplify and differentiate them: “Cloud computing interoperability is the ability for multiple Cloud providers to work together or interoperate. Cloud Compatibility and Portability answer to the question “how?”. Cloud Compatibility means application and data that work with the same way regardless of the Cloud provider, whereas Cloud Portability is the ability of data and application components to be easily moved and reused regardless of the provider, operating system, storage, format or API”[13].

Expressing the Open Grid Forum’s (OGF™) directions, Lee differentiates portability from interoperability [21]. According to his definition, interoperability means “being able to avoid “Cloud silos” that are non-interoperable since they are built on different APIs, protocols, and software stacks”. On the other hand, portability means “being able to perceive critical properties, such as performance, numerical stability and monitoring”.

The aforementioned definitions deduce that interoperability between different Cloud providers involves the transparent, automated exchange of code, data, applications and virtual servers. However, a more in-depth analysis of Cloud computing interoperability is necessary for
identifying the main dimensions of interoperability. Such an approach is presented in the next section.

2.1.2. Types of Cloud Computing Interoperability

This section provides an overview of different Cloud computing interoperability viewpoints. It attempts to illustrate the different interoperability classifications and to explore the characteristics/aspects of each category. The clear understanding and identification of the requirements that ensure interoperability is definitely the first step towards the standardization of Cloud computing platforms, APIs and services.

An approach in delimitating Cloud computing interoperability is presented by Sheth and Ranabahu [3], where Cloud computing interoperability is closely associated with the type of heterogeneity that arises during the interoperation of Clouds. Clouds interoperate to meet the needs of client applications using infrastructure, platforms or services coming from different Clouds. Sheth and Ranabahu recognize two types of heterogeneity; vertical and horizontal. Using the term “silo”, they refer to IaaS, PaaS or SaaS Cloud level.

Vertical heterogeneity emerges within a silo, i.e. when a customer needs to utilize services from different layers of the Cloud stack, but within the same silo. Horizontal heterogeneity depends on the level of the Cloud stack and therefore it is divided in three subcategories, dealing with the interoperability at IaaS, PaaS and SaaS level, respectively. It emerges when a customer opts to use simultaneously more than one services hosted at different providers or to change service provider, while also remaining in the same Cloud stack layer.

Furthermore, they describe portability as inversely proportional to automation across different layers (Figure 2). Since providers of higher layers offer customised services with limited set of configurations, automation increases but it negatively affects the portability and the interoperability of the system.

Moreover, besides Google App Engine, cloudControl, one of the trial partners of Cloud4SOA project, is also included in the providers offering PaaS solutions.

![Figure 2: Portability and automation across Cloud layers [3]](image)

ENISA focuses on the lock-in problem at each level of the Cloud stack separately, since each level deals with different services [4]. According to ENISA, IaaS computing offerings consist of
software and virtual machine (VM) metadata which are bundled together and distributed. Hence, the lock-in problem at the IaaS level depends on the infrastructure services utilized. Therefore, a customer using Cloud storage will not be impacted by not-compatible VM formats. He will be affected only by different features sets, which in this case are data characteristics, as well as semantics (terminology) which describe these features and vary significantly in the storage offerings.

In the same document, PaaS lock-in occurs at both the API (i.e. platform specific API calls) and the component level (i.e. a PaaS provider may offer a higher efficient back-end data store). Therefore, even if a compatible API is offered, the data may not be portable across PaaS offerings, as different data access models may exist. For security reasons PaaS environments often use heavily customized runtimes which also affect interoperability.

Lock-in problems are also identified, according to ENISA, in the SaaS layer. They occur at data and application level. Customer data is typically stored in a custom database schema designed by the SaaS provider. Most SaaS providers offer API calls to read data records, but lock-in can’t be prevented as they do not offer readymade data export routines. In such a case, the customer has to develop a program to extract its data and write it to file ready for import to another provider. On the other hand, SaaS providers usually develop custom applications tailored to the needs of their target market therefore switching between different providers may need re-writing of applications to interact with the new provider’s API.

Following a different categorization, Urquhart [22] claims that Cloud computing interoperability is an issue that arises in i) application/service, ii) management and iii) image/data level. In this categorization, PaaS and SaaS are both referred as applications. Application interoperability is the ability of developers to create loosely coupling applications which are platform independent, while management interoperability depends on APIs compatibility. Image or data interoperability is based on how a virtual server image, a Java application or a database is defined, in order to be deployed on another provider.

2.1.3. Standards for Cloud Computing Interoperability

There are several initiatives trying to address interoperability through standardized Cloud models and APIs. Standardization bodies, non profit groups and member operated organizations work on advancing Cloud computing interoperability standards, with the collaboration of academia, researchers, governments and vendors.

The Distributed Management Task Force (DMTF)\(^6\), a group dealing with standards that promote interoperable IT management among multi-vendor systems, tools and solutions, has introduced the Open Cloud Standards Incubator\(^7\), which aims to standardize the interactions among Cloud environments by developing Cloud management use cases, architectures and interactions. Recently, DMTF has introduced the Cloud Management Working Group (CMWG)\(^8\) to continue the work of Open Cloud Standards Incubator. CMWG will develop a set of prescriptive specifications delivering architectural semantics as well as implementation details to achieve interoperable management between service requestors/developers and service providers. The CMWG will propose a resource model that will capture the key artifacts identified by “Use Cases and Interactions for Managing Clouds” document [23] produced by the Open Cloud Incubator.

---

\(^6\) http://www.dmtf.org/home
\(^7\) http://www.dmtf.org/standards/Cloud
\(^8\) http://dmtf.org/sites/default/files/CloudManagementWGCharter.pdf
The document focuses on use cases, interactions, and data formats while describing how standardized interfaces and data formats can be used to manage Cloud environments.

The Open Cloud Manifesto\(^9\), an initiative supported by tenths of companies, including major software and infrastructure vendors such as IBM, SAP, Siemens and Telefonica, argues that Cloud computing should capitalize on open standards. It tries to address standards around security, integration, portability, interoperability, governance etc [24].

The Open Cloud Consortium (OCC)\(^10\) is a member operated organization supporting the development of reference implementations, benchmarks and standards for Cloud computing technology and also of frameworks for interoperation between different Cloud computing providers. One of OCC’s working groups, the Working Group on Standards and Interoperability for Large Data Clouds focuses on developing standards for the interoperation of Cloud data infrastructures, for example, standard interfaces for storage and computing power. This working group is also developing the MalStone\(^11\), a benchmark that provides on-demand computing capacity.

The Organization for the Advancement of Structured Information Standards (OASIS)\(^12\) sees Cloud computing as an extension of Service-Oriented Architecture (SOA) and network management models. The OASIS plans to develop Cloud models, profiles and extensions on existing standards, including a) security, access and identity policy standards, b) content, format control data import/export standards, c) registry, repository and directory standards, and d) SOA methods and models, network management service quality and interoperability.

The Open Group Cloud Work Group\(^13\) aims to create a common understanding among buyers and suppliers, indicating how enterprises of all sizes can include Cloud Computing technology in a safe and secure way in their architectures to improve cost, scalability and agility benefits. It implies the development of standard models and frameworks that eliminate the vendor lock-in problem. The Open Group has established several project towards this direction including Cloud business use cases, and artifacts, Cloud computing architectures, service-oriented Cloud computing infrastructure and security.

Similarly, the growth of a vibrant commercial marketplace for Cloud based services, where major buyers and sellers collaborate to define a range of common approaches, processes, metrics and other key service enablers, is the primary objective of TM Forum’s Cloud Services Initiative\(^14\). For this purpose, TM Forum has introduced the Frameworx Integrated Business Architecture, with the following components: Business Process Framework (eTOM)\(^15\), Information Framework (SID)\(^16\), Application Framework (TAM)\(^17\), and Integration Framework\(^18\). The eTOM framework defines an industry’s common process architecture for business and functional processes, the SID offers a common reference model that service providers, software providers, and integrators use to describe management information, the TAM framework provides a common language between service providers and their suppliers to describe systems and their functions, as well as a

\(^9\) http://www.openCloudmanifesto.org/index.htm
\(^10\) http://openCloudconsortium.org/
\(^11\) http://code.google.com/p/malgen/
\(^12\) http://www.oasis-open.org/
\(^13\) http://www.opengroup.org/Cloudcomputing/
\(^14\) http://www.tmforum.org/
\(^16\) http://www.tmforum.org/InformationFramework/1684/home.html
\(^17\) http://www.tmforum.org/ApplicationFramework/2322/home.html
\(^18\) http://www.tmforum.org/BestPracticesStandards/IntegrationFramework/4866/Home.html
common way of grouping them and the Integration Framework provides a service-oriented integration approach with standardized interfaces and supporting tools.

The scope of the Cloud Computing Interoperability Forum (CCIF)\(^\text{19}\) is to enable a global Cloud computing ecosystem, where the organizations are able to work together seamlessly. The creation of a common agreed upon framework/ontology, which enables the ability of two or more Cloud platforms to exchange information in a unified way, is of primary importance. The Unified Cloud Computing, the project that will implement the above characteristics, will develop an open and standardized Cloud interface for the unification of several Cloud API's (an API for all other APIs). Furthermore, this unified Cloud interface will use Resource Description Framework (RDF) for the description of semantic Cloud data models. In order to achieve its scope, CCIF focuses on building community consensus, exploring emerging trends, and advocating best practices/reference architectures for the purposes of standardized Cloud computing platforms.

The Global Inter-Cloud Technology Forum (GICTF)\(^\text{20}\) aims to promote the standardization of network protocols and interfaces through which Cloud systems interwork with each other, promote international interworking of Cloud systems, enable global provision of highly reliable, secure and high-quality Cloud services, contribute to the development of Japan’s ICT industry and the strengthening of its international competitiveness.

There are many efforts focusing specifically on interoperability issues at the IaaS level. The Open Cloud Computing Interface (OCCI)\(^\text{21}\), one of the OGF’s working groups, intends to deliver an API specification for the creation of a new interoperable API, interfacing IaaS Cloud computing facilities, while allowing customers, integrators, aggregators, providers and vendors interoperate by using the same context.

Meanwhile, the ETSI TC CLOUD is interested in interoperability solutions associated with both IT and Telecommunications, giving emphasis on the IaaS delivery model. It envisions a coherent and consistent general purpose infrastructure, which could support networked IT applications in business, public sector, academic and consumer environments. These applications and services need to interoperate based on global standards.

The Object Management Group (OMG)\(^\text{22}\) is committed to provide models that will solve complex business challenges, including those associated with Cloud computing services. “Cloud computing, which is primarily a business decision of operating expense vs. capital expense, fits well into our vision of Business Ecology, which is focused on the optimization of business processes through standards” explains Richard Mark Soley, Ph.D., chairman and Chief Executive Officer of OMG. Therefore, OMG will work to deploy applications/services on Cloud platforms that will support portability, interoperability and reusing.

In order to advance global interoperability, a Cloud Computing Working Group has been also formed by the Network Centric Operations Industry Consortium (NCOIC CCWG)\(^\text{23}\), a global, not-for-profit organization that will investigate ways to leverage Cloud technology to support global interoperability. NCOIC will develop operational and capability patterns that can enable their customers (in military, aviation, emergency response and cyber security) to achieve portability of information and services from Cloud to Cloud. For the qualitative and quantitative assessment of Cloud computing interoperability, the Systems, Capabilities, Operations, Programs

\[^19\] http://www.Cloudforum.org/
\[^20\] http://www.gictf.jp/index_e.html
\[^21\] http://occi-wg.org/
\[^22\] http://www.omg.org/
\[^23\] https://www.ncoic.org/technology/deliverables/frameworks
and Enterprise (SCOPE)\textsuperscript{24} model is used. Targeting in the development of a set of Cloud computing interoperability best practices, the NCOIC CCWG is working on the identification of all significant Cloud computing interoperability dimensions.

Cloud Industry Forum (CIF)\textsuperscript{25} is an industry body that has been formed to advance and advocate the adoption and use of Cloud-based services by businesses and individuals. By the means of a Code of Practice, CIF will provide transparency of Cloud services such that consumers can have clarity and confidence in their choice of provider. This will be achieved by a trusted and sustainable marketplace where consumers and suppliers of Cloud Services can meet each other.

Recently, a consortium of businesses, called The Open Data Center Alliance\textsuperscript{26}, has been launched by Intel. The alliance’ scope is to specify the future hardware and software requirements that lead to more open and interoperable Cloud and datacentre solutions, with Intel playing an advisory role within the alliance. Specifically, it will work to produce open standards as well as deliver an open, interoperable and secure Cloud environment that will empower the next generation of IT services.

With respect to Grids and how Cloud computing can enhance the current Grid technology, the Standards and Interoperability for e-Infrastructure Implementation Initiative (SIENA)\textsuperscript{27} aims to break down the interoperability barriers that impede the pervasive use of grids and Clouds. This will be achieved through the co-ordination and co-operation between national and pan-European e-Infrastructure initiatives, recently funded Distributed Computing Infrastructure (DCI)\textsuperscript{28} projects, policy bodies, as well as enterprises, targeting at evaluating and responding to specific needs. To this end, SIENA also focuses on best practices related on how industry and a global network of individuals are addressing standards and interoperability issues. This sharing of best practices is an important step towards pushing forward interoperability through interaction with standardization development organizations.

The StratusLab\textsuperscript{29} aims to enhance Grid infrastructures with Cloud Computing by incorporating virtualization and Cloud technologies into existing Grid infrastructures in Europe while improving the usability of distributed computing resources.

The VENUS-C\textsuperscript{30} aims to develop and deploy an industry-quality Cloud computing service for research and industry communities in Europe by offering an industrial-quality, service-oriented platform based on virtualization technologies facilitating a range of research fields through easy deployment of end-user services.

Table 1 summarized the aforementioned standardization bodies classified based on their focus. It is observed that most of the existing work on Cloud computing interoperability emphasizes on the IaaS and SaaS level. Moreover, semantic interoperability conflicts arise at all three Cloud levels, i.e. IaaS, PaaS and SaaS [3]. Although the semantic interoperability research in the IaaS and PaaS is still very immature, things are different at the SaaS level where significant work has been conducted by the semantic Web services research community.

\begin{footnotesize}
\begin{tabular}{l}
\textsuperscript{24}https://www.ncoic.org/apps/group_public/download.php/8504/SCOPE_MODEL_VER1.0.pdf \\
\textsuperscript{25}http://www.Cloudindustryforum.org/ \\
\textsuperscript{26}http://www.opendatacenteralliance.org/ \\
\textsuperscript{27}http://www.sienainitiative.eu/ \\
\textsuperscript{28}http://www.sienainitiative.eu/StaticPage/Projects.aspx \\
\textsuperscript{29}http://www.stratuslab.eu/doku.php \\
\textsuperscript{30}http://www.venus-c.eu/Pages/Home.aspx \\
\end{tabular}
\end{footnotesize}
Nevertheless, in the near future multiple standards are expected to emerge for all three layers [25]. Lee, the president of the OGF, admits that the Cloud computing standardization will primarily occur at the infrastructure level but promptly it will proceed to the next two layers [21].

| Table 1: Interoperability – Standardization Bodies Initiatives |
|-----------------|-------|-----|-----|
|                 | IaaS  | PaaS| SaaS|
| DMTF/CMWG       | ✓     |     |     |
| Open Cloud      | ✓     | ✓   |     |
| Manifesto       |       |     |     |
| OCC             | ✓     |     |     |
| OASIS           |       | ✓   |     |
| Open Group Cloud|       |     | ✓   |
| Work Group      |       |     |     |
| TM Forum        |       |     | ✓   |
| CCAF            | ✓     |     |     |
| GICTF           | ✓     | ✓   |     |
| OGF/OCCI        | ✓     |     |     |
| ETSI TC CLOUD   | ✓     |     |     |
| OMG             |       | ✓   |     |
| NCIC            |       |     | ✓   |
| CIF             |       |     | ✓   |
| Open Data Center| ✓     |     |     |
| Alliance        |       |     |     |

2.2. Cloud PaaS Interoperability

As it can be observed in Table 1, most of the existing standardization efforts mainly focus on the IaaS and/or the SaaS levels. The lack of standards for Cloud platforms adds to the lock-in to a specific PaaS provider [26, 27]. Hence, developers are bound to specific platforms, frameworks, development tools and APIs [28], and code or data migration requires significant effort and additional costs [29]. Therefore, developers seem to prefer developing their application from scratch on IaaS offerings rather than using the, certainly more efficient but less flexible, PaaS offerings to avoid the lock-in to a specific platform. This scepticism towards PaaS impedes its utilization.

Since a lot of work has been done towards IaaS, Cloud4SOA is going to investigate issues raised in the PaaS level. In particular, Cloud4SOA is mainly focused on facilitating PaaS-to-PaaS semantic interoperability, the rest of this sections offers a comprehensive analysis of the PaaS level characteristics as well as the of the existing semantic interoperability problems.

2.2.1. Platform as a Service in a nutshell

PaaS offerings offer a higher-level development platform, hiding the low-level details from the developer, such as the operating system, the load balancing, the data storage and access. This developing platform enables programmers to write code and create applications without worrying about software versioning or limited infrastructure resources. A PaaS vendor provides a runtime framework which enables developers to program the application’s code and execute it.
Furthermore, the runtime framework includes a set of development tools that assist the developer in writing the code (e.g. libraries and APIs that give access to the computational and storage resources) [30]. Specifically, a PaaS usually includes the following components [31].

1. An engine: a core including logic, needed to operate with the environment of the provider
2. Development tools: they are used by the developers to create applications
3. Testing tools: special tools created to test sources and developing systems
4. Third-party tools and services: software and services which supplement existent environment with additional features
5. Databases integration ability: developing applications are able to store information in databases
6. Middleware and programming interfaces offered to customers for developing applications

The current PaaS systems implement a number of features varying significantly among Cloud vendors. Therefore, PaaS systems can range from “stand-alone environments” to “add-on development facilities environments” [32]. Stand-alone environments are fully-equipped systems with all necessary functions for the efficient application development including testing, debugging and deployment functions. This type of systems does not depend on any SaaS applications in licensing, technical and financial issues. Another type of PaaS systems implements only the hosting services such as security and on-demand scalability, without development, debugging or test capabilities. On the other side, the add-on development facilities provide SaaS extensions to be customized by developers and users.

A different approach classifies PaaS systems according to the level of implemented functions in the lifecycle platforms [33]. Based on this categorization, “integrated lifecycle platforms” offer all facilities needed to develop applications from operating system to collaborative and version control tools. The “anchored lifecycle platforms” differ from integrated platforms in the underlying business software, with Salesforce.com being a representative example of this platform type. There are also some Cloud platforms that do not offer the corresponding tools which are necessary for the convenient management of all facilities. In such a case, several primary tasks like testing, service management, integration and configuration are carried out by external technologies named “technologies as a platform”.

Within Cloud computing platforms, four different development models can be identified: (i) application development platform model, (ii) virtual process platform model, (iii) virtual machine platform model, and (iv) PaaS access platform model. Each of them constitutes a different abstraction level of PaaS, offering different toolsets of functionality [34]. The application development platform model offers complete application development functionality, including data definitions, coding, testing, releasing as well as Web APIs integrated with the development platform. The virtual process platform model provides a virtual operating environment of a specific language, with specific Software Development Kit (SDK) programmable functionality and a local simulator as a debug tool. Developers program the applications locally, and then upload them to the platform. The virtual machine platform model is the lowest level of the Cloud computing environment development model providing developers with a basic virtual computer environment. This virtual machine model consists also the most common form of the IaaS solutions. The PaaS access platform model does not provide computing capability to developers, but offers access functions where “access” is the relationship between PaaS platforms and the third-party applications.

Ret Hat proposes a categorization of PaaS solutions where they vary from specialized environments that enable extensions to SaaS and offer components and frameworks for the solution of very specific classes of business problems to general-purpose middleware, which
are intended for development, deployment, and management of a large number of applications that address a wide range of business challenges [35].

<table>
<thead>
<tr>
<th>SaaS with extensions</th>
<th>Customize and extend the capabilities of a SaaS application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose-built PaaS</td>
<td>A framework that simplifies the development of a specific class of applications</td>
</tr>
<tr>
<td>PaaS tied to a single application paradigm</td>
<td>Provides general capabilities, but supports only one programming model or development/deployment environment</td>
</tr>
<tr>
<td>PaaS tied to a single Cloud</td>
<td>May provide general capabilities, but can be used only in the context of a single public Cloud or a single type of private Cloud infrastructure</td>
</tr>
<tr>
<td>Middleware hosted in the Cloud</td>
<td>Eases distribution of middleware across the organization, but adds no other value</td>
</tr>
<tr>
<td>General-purpose PaaS</td>
<td>Comprehensive, open, and flexible solution that simplifies the process of developing, deploying, integrating, and managing applications in public and private Clouds.</td>
</tr>
</tbody>
</table>

### 2.2.2. Cloud PaaS Interoperability Problems

Taking a closer look into the Cloud market, limited development tools are offered by the current Cloud providers, most of which use their proprietary runtime frameworks, programming languages and APIs supporting also on-premise toolsets and SDKs, while APIs come short to provide the underling middleware capabilities. These incompatibilities obstruct the uniform interoperation of PaaS providers and the consistent migration of the application code.

A discussion follows about the most important interoperability issues that emerge on the PaaS level, with emphasis on interoperability conflicts.

PaaS runtime frameworks vary significantly. There are runtime frameworks based either on traditional application runtimes or on 4GL (Fourth-generation programming language) and visual programming concepts. On the other hand, there are runtime frameworks with pluggable support for multiple application runtimes [36].

Furthermore, there is not a standard programming language of PaaS offerings. Google App Engine supports only Python and Java, while Force.com applications are developed on proprietary programming languages like Apex. There are several ways to overcome this incompatibility, e.g. cloudControl overcomes the lock-in problem sitting on top of a standard and open LAMP stack while Amazon offers Beanstalk to build Java apps on Amazon Web Services (AWS) with other programming languages and partnerships to follow [37].

Moreover, the runtime environments lack standards for SDK. As a result, the parts of the application code which rely on the SDK functionality should be redesigned when the developer chooses to move his application to another platform.

As discussed in section 2.2.1, a significant issue of the current PaaS ecosystem is the wide range of different types of PaaS services. For example, some platforms offer little more than a set of APIs on top of an elastic infrastructure, while others offer fully functional web-based Integrated Development Environment (IDE) and/or 4th generation programming language environments, allowing an easy creation of metadata-level mash-ups [38].
Data types and storing methods differ between Cloud platforms while most big vendors prefer offering their on-premise databases. For example, Microsoft Azure uses the SQL Azure Database while Google App Engine its on-premise DataStore. DataStore allows structured and semi-structured data transactions and has a simple query language, called GQL. However, DataStore lacks the functionality of a “full” database and GQL, being a restricted language similar to SQL, lacks the functionality of joins and complex aggregations. On the other hand, SQL Azure Database provides the features of a MS SQL Server database. However, it is restricted in size [39].

Conflicts arise also in the back-end functionalities of applications like accounting, billing, metering or advertising, since each PaaS provider introduces its own standards [2].

The aforementioned interoperability problems can be summarized in the following list:

- Lack of common/standardized Cloud PaaS APIs
- A great diversity in frameworks, languages, toolsets and SDKs (proprietary)
- Different levels of services (different types of PaaS solutions)
- Heterogeneous data types and storing methods
- Non-interoperable accounting, billing, metering and advertising services

### 2.3. Interoperable Cloud Computing Architectures

This section presents Cloud computing solutions aiming to address (semantic) interoperability. The proposed architectures are classified based on their origin into: 1) architectures introduced by industry in the context of commercial products and solutions, 2) architectures proposed by academia and 3) architectures proposed by R&D projects. Their analysis exposes the gaps and deficiencies that existing Cloud computing architectures suffer from. The conclusive discussion at the end of this section displays a correlation between the visions of the Cloud computing architectures and Cloud4SOA’s and depicts the actors identified in each of the architectures. Furthermore, it provides an overview of a generic architecture that covers the most commonly used architecture layers.

#### 2.3.1. Industry-driven Approaches

This section reviews the following Cloud computing architectures proposed by industry in the context of commercial solutions.

- eCloudManager [40];
- Aneka [41];
- Cloud Exchange Federated Cloud [42];
- SaaS-PF [43];
- Open PaaS [35];
- Red Hat Reference Cloud Computing Architecture [44];
- Cisco Reference Cloud Computing Architecture [45];
- IBM Reference Cloud Computing Architecture [46]; and
• Oracle Architecture for private PaaS [47].

2.3.1.1. eCloudManager

eCloudManager [40] aims to provide a Cloud computing solution that will support automated resource provisioning. It focuses on semantics and how this technology can assist the intelligent information management.

The eCloudManager Product Suite (Figure 3) aims at the management of enterprise Cloud environments. The bottom of the architecture incorporates the two dimensions of the information, the Data Center Resources and the Business Resources. The Data Center Resources are divided into the Hardware layer, the Virtualization layer and the Application layer. The Hardware layer consists of physical storage, network and compute infrastructure while the Virtualization layer, built on top of the Hardware layer, is made up of virtual resources such as hypervisors with the appropriate management capabilities (enabling virtual clustering, live migration etc.). The Application layer consists of applications and landscapes (enterprise systems) on top of the virtualized resources.

On top of the aforementioned layers, four complementary layers, named Intelligence, Infrastructure Management, Virtual Landscape Management and Self-Service, are built.

The Infrastructure Management layer provides monitoring and management services crossing the whole IT stack, like CPU and storage virtualization across different virtualization and storage providers. Its main features are the VM provisioning, the storage technologies as well as the error detection.

The Virtual Landscape Management layer offers interconnected, multitiered, multi-system enterprise applications as complete and ready-to-run landscapes.

The Self-Service layer constitutes an end-user oriented portal for template-based on-demand provisioning of applications for development, value prototyping, testing and production. An additional feature of this layer is a module for metering and billing.

The Intelligence layer makes use of innovative semantic technologies to integrate the available resources into a semantic data store. It is divided into the Data Integration layer, the Data Management layer and the Presentation layer.

The Data Integration layer relies on the concept of data providers. Abstracted from the details, a data provider is a component that extracts data from a single physical/logical resource, converts it into RDF and integrates the resulting RDF data into the central repository.

The Data Management layer contains the central repository where the provider data is written. Besides the repository, the layer provides components for search and intelligent, semantics-based information access. Another component of Data Management layer is the semantic wiki pages that are associated with the resources contained in the repository.

The Presentation layer, which is located on top of the Data Management Layer, provides a predefined set of widgets with varying functionality, e.g. offer support to the wiki pages, visualize the underlying data using charts and diagrams, navigate through the underlying RDF graph, and collaboratively annotate resources in the database using both semantic annotations as well as free-text documentation.
2.3.1.2. Aneka

Aneka PaaS supports provisioning of private Cloud resources ranging from desktops, clusters to virtual datacenters using VMWare, Citrix Xen server and public Cloud resources such as Windows Azure, Amazon EC2, and GoGrid Cloud Service. Aneka technology primarily consists of two key components:

- SDK (Software Development Kit) containing application programming interfaces (APIs) and tools essential for rapid development of applications. Aneka APIs supports three popular Cloud programming models: Task, Thread, and MapReduce; and
- A Runtime Engine and Platform for managing deployment and execution of applications on private or public Clouds.

Aneka’s architecture [41] is depicted in Figure 4.

The fabric services directly interact with the node through the Platform Abstraction Layer (PAL) and perform hardware profiling and dynamic resource provisioning.

The foundation services, the central component of Aneka middleware, provide a set of basic features on top of which each of the Aneka containers can be specialized to perform a specific set of tasks.

The execution services directly deal with the scheduling and execution of applications in the Cloud.

Additional services such as persistence and security cross the entire stack of services that are hosted by the Container.
On top of the Aneka middleware, the application level provides a set of different components and tools with the aim to:

1. Simplify the development of applications (SDK)
2. Port existing applications to the Cloud
3. Monitor and manage the Aneka Cloud

Figure 4: Overview of the Aneka’s Framework [41]

2.3.1.3. **Cloud Exchange Federated Cloud**

The vision of a federated Cloud computing environment which facilitates the scalable provisioning under variable conditions is termed InterCloud [42] (Figure 5). The main components of a federated architecture involve the client Brokering, the Coordinator services and the Cloud Exchange. A client initiates a Cloud broker in order to meet their needs, whereas Cloud Coordinators publish their services to the federation.

The Cloud Exchange acts as mediator bringing together service providers and customers. It aggregates infrastructure demands from the application brokers and evaluates them by supplying available resources, published by the Cloud Coordinators. The communication and transaction are assisted by SLA messages that are carried out in a secure and dependable environment [42].
2.3.1.4. **SaaS-PF**

SaaS-PF, Fujitsu’s Cloud solution, consists of an application platform and a resource platform [43] (Figure 6). The resource platform involves the basic infrastructure (IaaS) features and plays a foundation role supporting the overall environment. The application platform enables the development and execution of applications and consists of the application execution platform, the authentication/authorization management and the service operation management (dashboard). The execution capabilities are supported by a Java version/.NET version application execution platform. The authentication/authorization management includes authentication ID management, service tenant management and user management, whereas the service operation management involves log management, middleware and resource monitoring. Fujitsu plan to enhance their solution providing: application execution, shared, development, operational management and business platform functions.
2.3.1.5. **Open PaaS**

In [35], the design of a portable PaaS system is investigated. The main objective of a portable PaaS system is openness, and specifically, the open choice in development languages, tools and deployment. It also realizes a comprehensive set of middleware services and supports the complete application lifecycle management.

According to Red Hat, an open PaaS (Figure 7) incorporates two main capabilities: the PaaS services and the Cloud engine. The PaaS services are comprised of application and integration runtime services that application developers can use through APIs when they build their applications. These can be divided into application platform services and business/integration services.

The application platform services are basic PaaS services, supporting many component models (e.g. JMX, POJO, OSGi), programming APIs (Java Enterprise Edition, Spring Framework, Seam, Struts, Google Web Toolkit) and languages (e.g. Java, PHP, Ruby). They encapsulate messaging services to provide reliable transport for application and service integration and interoperability and a set of additional services such as transaction, Cloud-aware clustering, storage, data and web services. The PaaS integration services make integration capabilities available as PaaS services. They consist of integration, presentation, user interaction, rules and business process services. The business process services are responsible for modeling, workflow and orchestration of flow, and monitoring.

The Cloud engine is a set of management and provisioning services that will enable developers to create and manage their IaaS and PaaS resources.

![Figure 7: The design of an open PaaS system [35]](image)

2.3.1.6. **Red Hat Reference Cloud Computing Architecture**

The aim of the reference Cloud computing architecture (Figure 8) released by Red Hat (via JBoss Enterprise [44]) is to provide a roadmap of the open source software capabilities across a diverse set of application requirements. The reference architecture focuses on openness, flexibility, and scalability that enterprises need when develop and deploy their applications. Specifically, it realizes a broad set of middleware capabilities, including:
1. Application and services runtime services
2. Process management and service integration (SOA)
3. Data integration and business intelligence services
4. User interaction services (Portal Platform)
5. Systems management and monitoring
6. Integrated development tooling

![Figure 8: A reference architecture released by Red Hat [44]](image)

### 2.3.1.7. Cisco Reference Cloud Computing Architecture

Cisco’s reference architecture for Cloud computing (Figure 9) is made up of five architectural layers, connected via APIs and repositories [45].

![Figure 9: Cisco’s Cloud Reference Architecture [45]](image)

The bottom layer, named Technology layer, consists of three basic components: network, compute and storage. On top of the foundation layer, the Security layer is applied to the overall (end-to-end) architecture and provides data and resource access control, encryption and incident detection. The Service Orchestration layer is responsible for the integration of the lower layers to create a service for delivery. This layer is implemented with configuration repositories which store key information and facilitate mapping of the technology components to the service components. The Service Delivery and Management Architecture layer is the place where the infrastructure and service management functions take place, e.g. control functions that manage the data and the applications which are deployed and used, compliance and service-level
management functions that are responsible for contracting and enforcement of SLAs. The topmost layer is the Consumer-Facing layer, usually exposed via a portal solution. This is the layer which allows services to be defined, requested and managed by the end users.

2.3.1.8. IBM Reference Cloud Computing Architecture

IBM’s reference architecture (Figure 10) provides a comprehensive set of capabilities required to address the needs of end-users, providers and creators of Cloud services [46].

The creator’s capabilities allow (a) design and build, (b) store, (c) deployment, and (d) management of the entire lifecycle of “images”, where the “image” can include the IT resources, the operating system, the middleware and the applications.

The service consumer component provides capabilities that serve both the end-users and the operators to manage the infrastructure, such as ensuring that the images that can be accessed are defined in a catalogue.

The key capabilities of the reference architecture are defined in the service provider component. The bottom layer defines the capabilities of the virtualized infrastructure while the next layer provides middleware capabilities such as image deployment, security, workload management and high-availability. The middleware is built in order to deliver services and information according to well defined SOA and Information architecture. The lifecycle management plays a major role in IBM’s reference architecture. Specifically, the architecture provides tools to manage user requests (manage the self service requests, the lifecycle of images and the provisioning of images based on the request) and to handle qualities of service associated with the delivering images (availability, backup and restore, security and compliance, and performance management.). Virtualized resources and workloads demand also management. Furthermore, the usage and accounting management help define business and IT metrics, meter the usage of services and resources and accounting.

![Figure 10: IBM’s reference architecture [46]](image-url)
2.3.1.9. **Oracle Architecture for private PaaS**

A similar reference architecture is proposed by Oracle for private PaaS Clouds (Figure 11) [47]. The architecture enables different departments to access and develop their applications in the local Cloud. The physical infrastructure consists of servers, legacy systems such as mainframes, integrations, and database resources. The middleware layer is made up of application servers and technologies such as SOA, BPM, user interface (UI) technologies, identity and systems management. Upon this foundation, custom elements including shared components such as SOA services, BPM processes and self-service interfaces are located.

![Figure 11: A reference architecture for PaaS private Clouds released by Oracle [47]](image)

2.3.2. **Academia-driven Approaches**

The following research studies, undertaken by academia and research institutes, are focusing on the basic functionality of PaaS systems:

- The Cloud Development Stack model [48];
- Chappel’s three-partite model [49];
- The next generation Cloud architecture [50];
- Charlton’s Cloud computing reference architecture [51];
- The Cloud computing reference model [10];
- Sambyal et al. Cloud computing model [16];
- The adaptive PaaS architecture [26]; and
- The Cloud deployment model [52].

2.3.2.1. **The Cloud Development Stack model**

The Cloud Development Stack model [48] enables a cost-effective development in the Cloud while implements a number of PaaS requirements such as scalability, reliability, highly availability, flexibility in deployment, use interfaces and portability. The model provides an objective framework of what needs to be included by a perfect PaaS in order to develop and deliver efficient and effective Cloud solutions. The model and its core components, depicted in Figure 12, are summarized below.
The infrastructure layer includes the most basic PaaS capabilities such as hardware, software and the associated infrastructure upon which application development is executed.

The middleware layer offers the appropriate capabilities, including virtualization, security and multitenancy, which will enable the development of efficient Cloud computing applications. Additionally, the middleware layer can include services such as BPMS and other Workflow engines, Content Management Systems, Runtimes and Data Services.

On top of this, the development layer constitutes the main layer of PaaS model where the “real” development work is taking place. This is the most complex layer within the model, as it must include all the critical components/tools/languages/libraries required by developers to develop, test, refine and link their offerings.

Finally, the application layer supports two types of applications to be developed and deployed in the Cloud: Native Cloud Apps and Ported Cloud Apps.

The metering and analysis layer includes the “dashboards” and the analysis tools for PaaS usage and pricing. They provide the baseline management capabilities for developers to meter and match their PaaS usage to their PaaS SLAs and performance guarantees. They also provide an ability to see and track development cost models.

The administration layer offers tools that enable the discipline and the governance, making things work efficiently in the Cloud development. Key capabilities that should be included at this layer are the configuration management/version control system, data dictionary services and directory services.

2.3.2.2. Chappell’s three-partite model

A three-partite model, similar to the Cloud Development Stack, is presented by Chappell in Figure 13, where either on-premise or Cloud application platforms share the same functionality provided by foundation, infrastructure and application services [49].

The foundation service refers to the operating system and the software (e.g. libraries and storage) available on the machine hosting an application.

The infrastructure services provide remote storage, integration services or identity services.

The applications services encapsulate existing service-based applications into new composite applications.
2.3.2.3. The next generation Cloud architecture

The authors of [50] envision the next generation Cloud architecture where the management of physical resources will be decoupled from the virtual resource management. In such a system, there is a high need for a mediation layer that will ensure the allocation of resources between multiple applications.

The proposed mediation layer, depicted in Figure 14, constitutes the main PaaS layer and consists of five components including Infrastructure Service Fabric, the Distributed Services Assurance Platform, the Distributed Services Delivery Platform, the Distributed Services Creation Platform and the Legacy Integration Services Mediation.

In the bottom, the Infrastructure Service Fabric layer (Distributed Services Mediation Layer and Virtual Resource Mediation Layer) deals with the uniform provisioning of resources across the different Cloud applications. The Distributed Services Delivery Platform is a workflow engine that executes the application which is composed as business workflow while the Distributed Services Creation Platform offers the development tools that will enable applications to be composed, decomposed and distributed on the fly to virtual servers. Meanwhile, the virtual
servers can be automatically created and managed independent of the physical infrastructure by the Distributed Services Assurance Platform, which provides FCAPS-management (Fault, Configuration, Accounting, Performance and Security) abilities to the service developers through appropriate management APIs. In addition, the Legacy Integration Services Mediation provides integration and support for existing or legacy applications.

### 2.3.2.4. Charlton’s Cloud computing reference architecture

Charlton [51] proposes a Cloud computing reference architecture. Figure 15 illustrates the three core layers and the three sets of shared services that cross the core layers. The topmost layer is the Application Plane which contains the abstract elements of an application’s architecture: its components and connectors, their lifecycles, the relationships and the requirements of configuration and of the infrastructure, and any policies, constraints or preferences on the elements of the application. The Inter-Cloud Control Plane involves the network of distributed servers and agents that provide the automated management and configuration to applications that span multiple infrastructure Clouds. The Management Plane enables the programmable access to the elements of IT infrastructure that enable applications. Regarding vertical layers, the Federated Identity Services enable a multi-organization trust model across the application configuration, its control panel and into the management plane. The Administration and Access Control Services manage an organizational authority data structure across all layers. Lastly, the Search, Syndication and Aggregation Services enable efficient caching, searching and queuing of current/desired state across the three layers.

![Figure 15: Cloud Reference Architecture [51]](image-url)

© Cloud4SoA consortium
2.3.2.5. Cloud computing reference model

The Cloud Computing Reference Model (CC-RM) (Figure 16) constitutes a standardized process for modeling Clouds facilitating the process of Cloud modeling, development, planning and architecture [10]. The CC-RM is comprised of four supporting models, namely the Cloud Enablement Model, the Cloud Deployment Model, the Cloud Governance and Operations Model and the Cloud Ecosystem Model. The Enablement Model describes the fundamental technology behind the Cloud computing capabilities. The Cloud Deployment Model describes the range of Cloud deployment scenarios such as internal/private, external/public etc. The Cloud Governance and Operations Model identifies the governance, security, management and monitoring, operations and support requirements that will enable the appropriate management and security control. Lastly, the Cloud Ecosystem Model introduces the required environmental ingredients for developing and sustaining a Cloud ecosystem comprised of Cloud providers, consumers and intermediaries.

Figure 16: Cloud Computing Reference Model [10]

The Enablement Model of CC-RM focuses on ensuring the provisioning, the management and the offering of Cloud resources to consumers. The functionality of this layer includes a number of capabilities that are listed below:

1. Virtualization technology
2. SOA enablement technology
3. Billing and metering
4. Chargeback and financial integration
5. Load balancing and performance assurance
6. Monitoring, management, and SLA enforcement
7. Resource provisioning and management
8. Onboarding and offboarding automation
9. Security and privacy tools/controls
10. Cloud pattern enablement tools
11. Cloud workflow, process management, and orchestration tools

The Cloud Platform tier, one of the tiers of the Enablement Model, is comprised of two sub-tiers, named the Cloud Platform Middleware sub-tier and the Cloud Platform sub-tier (shown in Figure 17). It provides the core platform functionality for application development, hosting,
support, messaging and mediation capabilities. The Middleware sub-tier includes all middleware technologies and tools needed to build an application platform while the Platform/PaaS sub-tier represents pre-integrated Cloud and application platforms, which can be offered as a service (PaaS).

Figure 17: The Cloud Platform tier as part of the Enablement Model [10]

2.3.2.6. **Sambyal et al. Cloud Computing model**

Sambyal et al. [16] propose a Cloud computing model that consists of four layers: metadata, “the bits”, the deployment/configuration layer and the runtime orchestration layer combined with the service level policies (Figure 18).

Figure 18: A new Cloud computing model [16]

Metadata describes the manifest of the package as well as any other metadata required for processing the package such as the specification version, the application classification, etc. The “bits” are the software and the data which are delivered. This can be in any applicable format, such as Open Virtualization Format (OVF)\(^3\). The deployment/configuration layer contains the information required to successfully get the application up and running in the target Cloud. This layer could include a lot of information, such as server and storage configurations, network

\(^3\) [http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf)
connections as well as information concerning to acceptable pricing and billing terms. Last, the orchestration and the service level policies are required to handle the automated run-time operation of the application bits.

### 2.3.2.7. Adaptive PaaS Architecture

Rymer presents an adaptive PaaS architecture (Figure 19), which differs from a traditional PaaS as it removes development tools and frameworks from the PaaS stack and provides only packaging, deployment, distribution management, workload management, and resource virtualization services [26]. At the same time, developers achieve interoperability without having to use special APIs or languages to create their applications for Clouds since the adaptive PaaS adapts the conventional application code to the elastic scaling and multi-tenancy of Cloud architecture.

![Figure 19: An adaptive PaaS architecture [26]](image)

### 2.3.2.8. Cloud deployment model

A similar deployment model is proposed by Amedro et al. [52] (Figure 20). It achieves interoperability giving the users the capability to deploy the applications on different platforms without changing the source code. The main idea behind the proposed model is to remove processes related to the discovery of resources, the creation of remote processes and the data handling from the main application.

![Figure 20: Application and Deployment descriptor [52]](image)

The deployment model is based on two XML descriptors, named Application and Deployment descriptor. The Application descriptor depicts the application requirements, while the deployment descriptor defines the deployment processes. As a result, significant advantages
towards migration are achieved as the users can add a new resource provider, without changing the application code. On the same time, the definition of the deployment process happens once for each resource and can be reused for different applications.

2.3.3. R&D project-driven Approaches

This section reviews existing Cloud computing architectures that have been proposed by EU-funded R&D projects. We mainly focus on architectures that deal with Cloud computing (semantic) interoperability, even if this is not their primary objective. The following projects are considered:

- 4CaaSt [53];
- CumuloNimbo [53];
- Cloud-TM [53, 54];
- mOSAIC [53, 55];
- CONTRAIL [53];
- Vision Cloud [53];
- REMICS [53] [56];
- RESERVOIR [57];
- SLA@SOI [58];
- SITIO [59];
- NEXOF [60];
- Cloud@Home [61]; and
- SOA4All [62].

2.3.3.1. 4CaaSt

The 4CaaSt project\textsuperscript{32} aims to create a PaaS platform that will provide a high level of abstraction regarding application hosting. To this end, it proposes a set of built-in programming libraries and common facilities beyond what is offered by State of the Art PaaS Clouds, facilitating the development and the monitoring of the applications. This way, an attractive business ecosystem can be created, with an active community of users and developers [53]. Figure 21 provides an overview of the proposed architecture.

The portal that will be provided by the 4CaaSt-compliant Cloud will use an Application Blueprint descriptor to determine the application components, SLAs, business rules, third part services used, as well as the exported services. A Runtime Execution Container will be set up that will host the application components. These components will be distributed among the available Runtime Execution Containers by the 4CaaSt Cloud platform [53].

\textsuperscript{32} http://4caast.morfeo-project.org/
2.3.3.2. **CumuloNimbo**

CumuloNimbo\(^{33}\) aims to provide a scalable PaaS Service which will enable secure and un-partitioned data transactions resulting in consistent applications and at the same time ensuring the independent and optimized use of resources at a minimum cost. The CumuloNimbo subsystems will be self-healing, automatically repairing themselves in the case of technical failures in order to avoid problems during the service provisioning [53]. Figure 22 provides an overview of the CumuloNimbo architecture.

![Figure 21: Architecture proposed by the 4CaaSt project [53]](http://cumulonimbo.eu/)

![Figure 22: Architecture of the CumuloNimbo Project [53]](http://cumulonimbo.eu/)

\(^{33}\) [http://cumulonimbo.eu/]
2.3.3.3. **Cloud-TM**

Cloud-TM\(^34\) focuses on the design, the development and the assessment of an innovative middleware platform that will facilitate the development of applications and will provide Cloud-based services as resources at minimum costs. The platform will also be able to autonomously acquire or release resources from the Cloud, and accordingly regulate its consistency mechanisms to maximize performance and efficiency [53] [54]. The architecture of the platform can be seen in Figure 23. The key components are: the Distributed Software Transactional Memory (DSTM), the Distributed Storage System (DSS) and the Autonomic Manager (AM).

The DSTM represents the transactions in an abstract level in order to break through the limitations of using a single machine and to simplify parallel programming. Moreover, DSTM synchronizes different actions in one phase in order to amortize communication outflows across different memory accesses.

The DSS stores both the user level application data and long-lived middleware data. This component describes an API that will summarize several concerns of persistence in the Cloud, such as scalability and data partitioning.

The AM will serve as a feedback control component by deriving workload information from all the architecture’s layers. Moreover, this component will be responsible for the determination of the resources regulation criteria [53].

![Figure 23: Architectural view of the Cloud-TM platform [53]](image)

2.3.3.4. **mOSAIC**

The mOSAIC\(^35\) project aims at the development of an open-source platform that will be used as a means of communication between applications and Cloud services. In the long run, it is expected to serve as an environment for competition between Cloud providers, an aspiration shared by Cloud4SOA. The applications will be able to retrieve service requirements requested by

---

\(^{34}\) [http://www.cloudtm.eu/](http://www.cloudtm.eu/)

\(^{35}\) [http://www.mosaic-project.eu/](http://www.mosaic-project.eu/)
their users and send them to the platform via a consistent API. In turn, the platform, using a multi-agent brokering mechanism, will perform a search of services that match these requirements and send the results back to the platform [53]. The mOSAIC platform (Figure 24) has two main parts: the Resource Broker and the Application Executor [55].

The Resource Broker is responsible with resource negotiation and booking and consists of two sub-systems, namely the Client interface and the Cloud agency. The first describes the application resources needs and requests supplementary resources by the Application Executor. The second one uses a set of tools (a monitor, a negotiator, a mediator, a service registry, a client semantic engine and provider semantic engines), a Cloud ontology and QoS parameters to validate the application specifications and generates the SLAs.

The Application Executor, which is in charge of application execution based on specific SLAs, is comprised by several sub-systems, like the API Execution Engine, the Providers wrappers and the Resource manager. The API Execution Engine is the user's API for accessing the physical resources and the Virtual Cluster which includes the booked resources. The Providers wrappers are special connectors ensuring a uniform interface to the Clouds resources available in resource contract. Lastly, the Resource manager ensures resource availability and management, including Resource scheduler and Resource monitor, and handling supplementary resources request.

Figure 24: Platform architecture of the mOSAIC project [53]

### 2.3.3.5. CONTRAIL

The vision of the EU-funded CONTRAIL project is that any organization should be able to be both a Cloud provider when its infrastructure is not used at its maximum and a Cloud customer in periods of peak activity [53]. This way, cooperation and resource sharing over Cloud federations will be supported by the means of standardized interfaces. To this end, an open

---

source system will be designed, implemented, evaluated and promoted in which resources that belong to different operators are integrated into a single homogeneous Federated Cloud that users can access seamlessly. CONTRAIL will leverage and extend the results of the XtreemOS\(^{37}\) FP6 project, which builds a Linux-based operating system to support Virtual Organizations for next-generation Grids. Moreover, it will provide efficient vertical integration of PaaS and IaaS, QoS integration within infrastructure and comprehensive offering for Cloud platforms.

![Figure 25: Integrating multiple independent Clouds into a Federated Cloud](53)](53)

### 2.3.3.6. Vision Cloud

VISION Cloud\(^{38}\) will specify the architecture of a Cloud-based infrastructure (Figure 26) in order to build a reference implementation on open standards and new technologies. This way, a framework will be developed that will flexibly distribute data-centric storage services, overcoming the data lock-in problem and guaranteeing secure data interoperability \([53]\). To succeed this, an object data model, computational storage, content-centric access, comprehensive data interoperability, and QoS and security guarantees will play a central role in the entire infrastructure.

---


\(^{38}\) [http://www.visionCloud.eu/](http://www.visionCloud.eu/)
2.3.3.7. **REMICS**

REMICS\(^{39}\) will develop tools for model-driven reuse and migration of legacy applications to loosely coupled, interoperable Cloud services [53] [56]. For this purpose, REMICS will extend existing language to address the specific architectural patterns and model driven methods for architecture migration, and to cover specificities of service Clouds development paradigm.

In particular, the PIM4Cloud Computing, model-driven Service Interoperability and Models@Runtime extensions are intended to support the REMICS methodology for service Cloud architecture modeling. Moreover, the project focuses on open source meta-models while it

\(^{39}\) [http://www.remics.eu/](http://www.remics.eu/)
is actively involved in the standardization process of the related standards for Cloud computing, business models, SOA, service interoperability, knowledge discovery, validation and managing services.

2.3.3.8. **RESERVOIR**

RESERVOIR aims at the development of an innovative service-oriented infrastructure that will facilitate the dynamic interoperability of Cloud providers for the reliable delivery of services as resources. This will lead to a collection of technologically independent IT utilities that will be available on demand, increasing the competitiveness of the EU economy. RESERVOIR supports vertical as well as horizontal interoperability across the different architectural layers. According to the project’s architecture, the Cloud providers will describe their requirements in the same language, enabling their interoperation. Thus, service providers will be able to choose among the RESERVOIR Cloud providers, depending on their needs, realizing their participation in the Cloud provisioning market [57].

The components of the RESERVOIR’s architecture (Figure 28) are: the Service Manager, the Virtual Execution Environment Manager (VEEM) and the Virtual Execution Environment Host (VEEH).

The Service Manager represents the highest level of abstraction. This component interacts with the service providers to receive their Service Manifests, negotiate pricing, and handle billing. The information retrieved from the manifest enables the Service Manager to allocate VEEs and their associated resources by interacting with the Virtual Execution Environment Manager in order to deploy and provide the service application. Moreover, it monitors the application deployment in order to ensure SLA compliance i.e. by adjusting their capacity.

---

40 http://reservoir-fp7.eu/
The Virtual Execution Environment Manager interacts with the Service Manager, VEE Hosts, and other VEE Managers from associated sites. This component retrieves the limitations determined by the Service Manager and places accordingly the necessary VEEs into VEE hosts.

A Virtual Execution Environment Host has complete control over VEEs and their resources and it realizes the Service Manager’s IT management decisions into a diverse set of virtualization platforms. Moreover, VEEHs support transparent VEE migration to any compatible VEEH in a RESERVOIR Cloud [57].

2.3.3.9. SLA@SOI

SLA@SOI[^41] aims to develop a service-oriented infrastructure that will allow the exchange of IT services with flexibility and well defined conditions and costs. This infrastructure will lead towards the evolution of a service-oriented economy with formally specified SLAs and automated transactions between services [58]. Figure 29 illustrates the procedure of the service provision as addressed by the project. The main components identified are: the Business Manager, the SLA Managers, the Service Manager and the Monitoring and Adjustment Management System.

The Business Manager has overall control of all interactions with customers and providers and carries out the necessary procedures for the successful sale of the services to the customers.

The SLA Managers deal with all SLA related issues. There are three types of SLA managers, namely: the Business SLA Manager, the Infrastructure SLA Manager and the Software SLA Manager.

All types of SLA Managers can act as “service customers”, where they negotiate SLAs with other SLA Managers, either within the same framework, or with third party service providers. However, when they can act as “service providers” they can still negotiate SLAs with Managers within the same framework, but only Business SLA Managers can interact with customers outside of the framework by settling on an SLA and sending reports on the SLA status back to the customers.

A Service Manager communicates with the SLA Managers and provides them with management functions. This way, the SLA Managers are able to control the services instances. Moreover, Service Manager holds data such as the dependencies of a service with other services, and the implementations supported by the service.

Last, the Monitoring and Adjustment Management System contains Manageability Agents which support the actual configuration and management of service instances. This component is responsible for the overall control and management of the service instances across the software and infrastructure layer [58].

[^41]: http://sla-at-soi.eu/

© Cloud4SoA consortium
SITIO\textsuperscript{42} focuses on the creation of a framework and an architecture that will ensure the brokers’ access to business and Cloud computing services with a minimum cost. Based on these, a platform will be developed that will serve as a means to achieve reliable, secure and cost-efficient interoperability between heterogeneous applications, through the combination of concepts such as SaaS, Semantics, Business Process Modeling and Cloud Computing \cite{59}. Figure 30 depicts the basic components of the project’s architecture.

Four basic elements are identified by the SITIO architecture, namely User interface, Process services, Business services and Metadata services.

The user interface allows different users to access the SITIO platform and use and manage their accounts and applications. Software developers and IT providers visit the interface in order to deploy their applications, whereas the final users can search in the applications repository and subscribe to the ones they’re interested in.

The Process and Business services layers provide the ability to run arbitrary web services in the Cloud computing platform. To this end, distribution, load balancing and data persistence services are added to a given application transparently. However, to ensure this functionality, some restrictions on the services that run on top of the platform have to be imposed (e.g. specifying the programming language the application should be built on).

The distribution of an application is ensured by uploading it on one of the cluster’s synchronized application servers.

The Metadata Services layer uses ontologies for the semantic description of the available services. The annotators of the services are the Web services developers themselves, since they fully comprehend the methods of the services they develop \cite{59}.

\footnote{\url{http://www.eurekanetwork.org/project/-/id/4989}}
2.3.3.11. **NEXOF**

The NEXOF\(^43\) Reference Architecture is an open, coherent, consistent and comprehensive set of concepts and specifications. It can be used as a prototype for system architects to design architectures of service-based software systems which, in turn, provide solutions to a well defined set of requirements. Its main goal is to combine selected innovations in the area of service-oriented architectures and technologies to facilitate the implementations of interoperable service environments [60].

The NEXOF Reference Architecture consists of the following main elements:

- Architectural Guidelines and Principles that include principles underlying the construction of the framework as well as the set of reference properties associated with each of the components and patterns in the reference architecture. Furthermore, guidelines for the instantiation of a specific system architecture according to its requirements are incorporated

- The Reference Model that depicts the key entities that establish service-based systems as well as the relationships between them

- The Glossary that defines the terms used across the reference architecture

- The Reference Specifications that consist of the following elements:
  - Pattern Ensemble that describes the different ways in which functionalities can be achieved by associating components and other patterns
  - Standards Catalogue that describes the standards referred to in the reference architecture
  - Components Catalogue that groups both, abstract descriptions of components as well as product-or software-based components

\(^43\) [http://www.nexof-ra.eu/](http://www.nexof-ra.eu/)
2.3.3.12. **Cloud@Home**

The Cloud@Home project is funded by the French National Research Agency and aims at the creation of heterogeneous hardware for consistent Clouds. One of the primary goals of the project is the provision of a Cloud infrastructure which will allow the distribution of resources and services throughout the Cloud. Furthermore, the project aims the development of commercial Clouds as the basis of an open market where services are exchanged between users in a pay-per-use way [61]. To this end, we examine the basic architecture proposed by the Cloud@Home project, which is shown in Figure 32, while Figure 33 presents the configuration of the system.

The Frontend layer of the architecture manages the resources and services from the global Cloud system’s perspective. It is used for the translation of the end-user requirements into physical resources’ demand. It is also responsible for the negotiation, monitoring and adjustment of the SLAs in commercial Clouds. Therefore, this layer implements the interoperability among Clouds while also checking for the reliability and availability of services and requesting further resources and services to other Clouds when necessary.

There are three different available ways of accessing the Cloud, as shown in Figure 32, namely through the Cloud@Home frontend client, the Web 2.0 user interface and the low level Web interface (directly specifying REST or SOAP queries).

The Virtual layer is responsible for the virtualization of physical resources. This way, the end-users have a homogeneous view of the Cloud’s services and resources. One of the tools provided by the Virtual layer to the Frontend layer is the execution service, which is used for the creation and management (e.g. migration, replication) of VMs. This service from the end-user point of view is seen as a set of VMs. The second tool provided by the Virtual layer is the storage service,

---

44 http://Clouds.gforge.inria.fr/pmwiki.php

© Cloud4SoA consortium
which is seen from the end-user point of view as a remote disk. This service implements a distributed storage system across the storage hardware resources of the Cloud.

The Physical layer provides to the Virtual layer physical resources for the implementation of the execution and storage services, as well as mechanisms and tools for the local management of these resources.

![Figure 32: Basic architecture of the Cloud@Home project [61]](image)

![Figure 33: Configuration of the Cloud@Home system [61]](image)
2.3.3.13. **SOA4All**

SOA4All\(^{45}\) aims at realizing a world where a massive number of parties are exposing and consuming services via advanced Web technology. The main objective of the project is to provide a comprehensive framework that integrates complementary and evolutionary technical advances into a coherent and domain-independent service delivery platform (Figure 34)[62].

The SOA4All Studio delivers a fully Web-based user front-end that enables the creation, provisioning, consumption and analysis of services that are published to SOA4All. It consists of three subcomponents that target the three different service management tasks: provisioning at design time, consumption, and analysis at runtime.

The SOA4All Distributed Service Bus is the infrastructural backbone around which all the SOA4All components communicate and collaborate by combining Semantic Spaces and Enterprise Service Bus.

The SOA4All Platform Services is the group of services that provide the basic SOA4All functionality and activities, such as Service Ranking and Selection, Service Discovery, Service Adaptation, Service Composition, Service Execution, and the Reasoning Engine.

The Business Services (3rd party Web services and light-weight processes) are the actual services provided by the final users. The SOA4All framework aims to be technology agnostic and less intrusive as possible.

![Figure 34: The SOA4All Architecture [62]](image-url)

---

\(^{45}\) http://www.soa4all.eu/

© Cloud4SoA consortium
2.3.4. Discussion

Table 3 positions the architectures described in sections 2.3.1, 2.3.2 and 2.3.3 with respect to the main following key characteristics (that derive from the Cloud4SOA vision): the Targeted Cloud level; the Use of Semantics; Cloud Computing Interoperability; Data and Application Portability; and User-centricity.

As shown in Table 3, other efforts, similar to Cloud4SOA, try to address the issues of application and data portability and interoperability (not necessarily focusing on semantics) at the PaaS level. For example, Elastra employs semantics to deal with data integration and portability.

InterCloud describes an IaaS environment, that, similar to Cloud4SOA, aims to investigate functionalities such as the collection of the user’s requirements and the provision of services adapted to these requirements.

Several characteristics and functionalities included in the Cloud Development Stack model are particularly interesting for Cloud4SOA; especially the management abilities that are governed through the metering and analysis layer and the administration layer. Cloud4SOA will monitor closely the work of 4CaaSt, CumuloNimbo and CONTRAIL since they also focus on the PaaS level. mOSAIC and RESERVOIR allow Cloud providers to describe their requirements in the same language, ensuring their interoperability. They also enable users to express their requirements and find the best matching Cloud services. The models for describing Cloud resources (especially the mOSAIC Cloud ontology) and the matchmaking algorithms they use will be studied by the Cloud4SOA consortium in the context of WP2. VISION Cloud, SITIO and REMICS ensure application and data interoperability and portability. In particular, REMICS extends the current models and standards in the domain of Cloud computing, SOA and service interoperability while SITIO uses semantics which are two of the points that Cloud4SOA will dig into mainly in the context of WP2. Cloud4SOA will capitalize on the design principles set by NEXOF-RA and SOA4All. Similarly, the results SLA@SOI regarding service governance and negotiation will be considered, especially in the context of WP4 and WP5. Furthermore, most of the discussed architectures rely on user-centric principles which facilitate the direct access of users to their data and services through user-friendly UIs, thus reducing users’ dependencies on working on a specific operating system.

<table>
<thead>
<tr>
<th>Cloud4SOA drivers</th>
<th>Targeted Cloud level</th>
<th>Use of Semantics</th>
<th>Cloud Computing Interoperability</th>
<th>Data and/or Application Portability</th>
<th>User-centricity</th>
</tr>
</thead>
<tbody>
<tr>
<td>eCloudManager [40]</td>
<td>IaaS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aneka [41]</td>
<td>PaaS</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>InterCloud [42]</td>
<td>IaaS</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Fujitsu SaaS-PF [43]</td>
<td>PaaS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open PaaS [35]</td>
<td>PaaS</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Related projects vis a vis Cloud4SOA
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat [44]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Cisco [45]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>IBM [46]</td>
<td>All Cloud stacks</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Oracle [47]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Cloud Development Stack model [48]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Three-partite model [49]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Next Generation Cloud architecture [50]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Elastra [51]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Cloud Computing Model [16]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Adaptive PaaS Architecture [26]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Cloud Deployment Model [51]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4CaaS [53]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>CumuloNimbo [53]</td>
<td>PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Cloud-TM [53]</td>
<td>IaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>mOSAIC [53]</td>
<td>IaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>CONTRAIL [53]</td>
<td>IaaS &amp; PaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>VISION Cloud [53]</td>
<td>IaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>REMICS [56]</td>
<td>SaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>RESERVOIR [57]</td>
<td>IaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>SLA@SOI [58]</td>
<td>IaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>SITIO [59]</td>
<td>SaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>NEXOF [59]</td>
<td>-</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Cloud@Home [61]</td>
<td>IaaS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>SOA4All [61]</td>
<td>-</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
2.4. **Semantic Models for Cloud Computing**

This section documents the existing Cloud service models and explicates what a service can be in the Cloud, taking into account the related literature. Moreover, the existing Cloud resource models are identified and reviewed. A presentation of the different APIs that are used by Cloud implementations follows, classifying their main functions and concepts. Lastly, Cloud metamodels and concepts (including relationships and attributes) used for the modeling of Cloud platforms end this section. The conclusive discussion displays several research aspects and specifies a model of a common Cloud API based on reviewed literature.

**2.4.1. Cloud Service Models**

The following Cloud service modeling efforts have been reviewed:

- The Cloud Computing services taxonomy [19];
- The Cloud Computing services ontology [63];
- Intel Cloud Computing services taxonomy [64];
- Rimal et al. Cloud Computing Services Taxonomy [65]; and
- The Cloud computing stack [66].

**2.4.1.1. Cloud Computing Services Taxonomy**

A taxonomy of Cloud Computing services is introduced by [19]. The services of the taxonomy are:

i. **Infrastructure as a Service (IaaS).** It enables the provision of VMs to consumers

ii. **Platform as a Service (PaaS).** It allows the consumer to deploy onto the Cloud infrastructure applications

iii. **Software as a Service (SaaS).** It facilitates the usage of the provider’s applications running on the Cloud infrastructure by the consumer

**2.4.1.2. Cloud Computing Services Ontology**

An ontology for Cloud Computing services is proposed by [63]. The ontology identifies the main categories of a Cloud, as depicted in Figure 35a, but it also identifies their main subcategories, shown in Figure 35 b, c, d. More specifically, these subcategories are:

- Infrastructure as a Service. It can be a Computing as a Service (CaaS), an InfraSoftware, or a Computing Resources and Data as a Service (DaaS)

- Platform as a Service. It can be a Scalable Infrastructure, an Application Hosting or a Backup/recovery and Data security

- Software as a Service. It can be a Network, Software or a Database and System administration
2.4.1.3. **Intel Cloud Computing Services Taxonomy**

Intel proposes a taxonomy [64] for Cloud computing services in order to classify the entire breadth of existing Cloud technologies. The Cloud services and their classification are presented in Figure 36. The classification has two levels. At the first level the main Cloud Service categories are presented (i.e. Software as a Service, Infrastructure as a Service, Platform as a Service and Service as a Service), while at the second level each main category is further analyzed (e.g. the PaaS level is divided in instances like Business intelligence as a Service, Application Development as a Service, Web Hosting etc.).

![Figure 35: Cloud Ontology [63]](image)

2.4.1.4. **Rimal et al. Cloud Computing Services Taxonomy**

Another taxonomy for Cloud computing services is introduced by [65]. It distinguishes four different categories of services in the Cloud. The SaaS level uses common infrastructure to provide software applications to multiple customers simultaneously. The PaaS level provides a platform to the developers for the building and deployment of Cloud applications. The IaaS level delivers computer Infrastructure as a Service. Lastly, the Hardware as a Service (HaaS) level

![Figure 36: Cloud Computing Taxonomy by Intel [64]](image)
allows the client to use hardware as a pay-as-you-go service that scales up and down to meet his needs. The taxonomy is presented in Figure 37.

![Figure 37: A Cloud Computing Taxonomy](image)

### 2.4.1.5. Cloud Computing stack

A Cloud computing stack is presented by [66]. The stack includes 11 concepts:

1. Storage-as-a-service: disk space on demand
2. Database-as-a-service: usage of the services of a remotely hosted database
3. Information-as-a-service: consumption of information, remotely hosted, through an interface
4. Process-as-a-service: remote resources that can bind many resources together and create business processes
5. Application-as-a-service: any application delivered over the platform of the Web to a user
6. Platform-as-a-service: application development, interface development, database development, storage, and testing
7. Integration-as-a-service: a complete integration stack from the Cloud, including interfacing with applications, semantic mediation, flow control, and integration design
8. Security-as-a-service: security services remotely over the Internet
9. Management/governance-as-a-service: management of one or more Cloud services
10. Testing-as-a-service: testing local or Cloud-delivered systems using testing software and services that are remotely hosted
11. Infrastructure-as-a-service: remote access of computing resources

Figure 38 presents an overview of the Cloud Computing stack.
2.4.2. **Cloud Resource Models**

The following Cloud service modelling efforts have been reviewed:

- Han & Sim Cloud resource model [63];
- Haase et al. Cloud resource model [40];
- Oracle Cloud resource model [67];
- DeltaCloud resource model [68];
- OCCI Cloud resource model [69]; and
- Amazon Cloud resource model [70].

2.4.2.1. **Han & Sim Cloud resource model**

The ontology proposed by [63] also contains the Cloud resource model. For each subcategory of Figure 35b, 35c, 35d, it defines the resources needed. For example, the subcategory Application_Hosting of the PaaS level requires a programming language i.e. PHP, Java, Python etc.

An ontology for Cloud Computing Resources that can be used across heterogeneous Cloud providers is proposed in [71]. The main concepts of the ontology are the Network Bundle, the Processing Bundle and the Storage Bundle. The model is presented in Figure 39.
2.4.2.2. **Haase et al. Cloud resource model**

A conceptual model for Cloud computing resources in the form of an ontology is described by [40]. This model abstracts from vendors specific representations, data sources and management APIs. The major subareas of the model are the Storage Infrastructure, the Compute Infrastructure, the Application-level and the Business-level resources. The model is presented in Figure 40.

![Figure 40: Cloud Resource Model [40]](image)
2.4.2.3. **Oracle Cloud resource model**

Oracle proposes a Cloud resource model [67] that is also used by the Oracle Resource Model API. The model represents the Cloud resource hierarchy and the relations among the resources. For example, a Cloud has many Virtual Data Centers (VDC), where each VDC has many Virtual Networks (VNet) and finally each VNet has many Network Interfaces. The model is presented in Figure 41.

![Figure 41: Oracle Cloud Resource Model [67]](image)

2.4.2.4. **DeltaCloud resource model**

The DeltaCloud API [68], which is further analyzed at a later section of the document, is based on a resource model and depicted in Figure 42. The main concepts of the model are:

1. The hardware profile, which defines aspects such as local disk storage, available RAM, and architecture.
2. The image, which is a platonic form of a machine, where images are not directly executable, but are a template for creating actual instances of machines.
3. The instance, which is a concrete machine realized from an image.
4. The realm, which can represent different datacenters, different continents, or different pools of resources within a single datacenter.
2.4.2.5. **OCCI Cloud resource model**

OCCI [69] is a specification for remote management of Cloud computing infrastructure, allowing the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. OCCI proposes a Cloud resources model. The model’s concepts are the server, the storage and the network. These concepts as well as their relations are shown in Figure 43.

![OCCI Resource Model](image)

**Figure 43: OCCI Resource Model [69]**

2.4.2.6. **Amazon Cloud resource model**

Amazon is one of the leading companies at the area of Cloud computing platforms. The resource model that underlies the Cloud services offered by Amazon is presented at Figure 44 [70]. The concepts of the model are:

1. The image that is a predefined way to instantiate a VM.
2. The instance type that defines the instance characteristics (i.e. CPU, RAM, hard disk) as standard templates.
3. The Instance that is the realization of an image using an instance type.
4. The storage that denotes the component where the data are stored.
5. The network that connects many instances.
6. The IP address that is static IP addresses which can be related to any instance.
7. The availability zone that is a distinct location where the Cloud infrastructure is located. Each availability zone is insulated from failures in other Availability Zones.

![Diagram of Amazon Resource Model](image)

Figure 44: Amazon Resource Model

### 2.4.3. APIs for Cloud Computing

There are several APIs that are used by existing Cloud implementations. These can be divided into IaaS (section 2.4.3.1) and PaaS APIs (section 2.4.3.2) depending on the functionality they provide. A new kind of APIs has recently appeared focusing on the management of other APIs (section 2.4.3.3).

#### 2.4.3.1. IaaS APIs

IaaS APIs offer infrastructure-specific features. The following IaaS APIs have been documented:

- Amazon IaaS API [70];
- Rackspace API [72];
- GoGrid API [73];
- vCloud API [74];
- Oracle Cloud Resource Model API [67];
- ElasticHost API [75]; and
- FlexiScale API [76].

##### 2.4.3.1.1. Amazon IaaS API

The Amazon API [70] is widely used not only by Amazon Web Services but also by other open source Cloud management tools, such as Eucalyptus[^46], openNebula[^47] and Nimbus[^48]. The API is

[^46]: http://www.eucalyptus.com/
[^47]: http://www.opennebula.org/
[^48]: http://www.nimbusproject.org/
based on the Amazon Resource Model presented at Section 2.4.2. Three storage components are offered, namely EBS, S3 and SimpleDB, that differ in the way they store the data and in their usage (e.g. EBS can be used as an attached disk to an image while S3 and SimpleDB are simply used as independent storage services). Moreover, the API offers the capability to create virtual networks that serve as Virtual Private Clouds (VPC). The protocols used by the API are the SOAP and the REST. The main functions offered by the concepts of the API are presented in Figure 45.

2.4.3.1.2. Rackspace API

Rackspace [72] is a commercial API that supports the image and the instance (called Server) elements. Rackspace offers standard templates, named flavors, to create instances. It also offers the container as a storage unit that is similar to the notion of a folder used by operating systems. Furthermore, this API offers the capability to assign IP addresses to instances and create virtual networks. Rackspace uses the REST protocol. The main functions offered by the concepts of the API are presented in Figure 46.
2.4.3.1.3. **GoGrid API**

GoGrid [73] is a commercial API that supports the image and instance (called Server) elements. GoGrid does not offer standard templates to create instances, but the instances are created manually (i.e. CPU, RAM, HD). This API does not support storage, although the user can manage it through the Web interface. It offers the capability to assign IP addresses to instances and, therefore facilitates the creation of virtual networks. Finally, this API uses a concept named load balancer that is a server responsible to distribute workload across the available instances. The protocol used by this API is the REST. The main functions offered by concepts of the API are presented in Figure 47.

![Figure 47: The GoGrid API](image)

2.4.3.1.4. **vCloud API**

The vCloud API [74] constitutes an effort to develop a standardized API. It was submitted to the DMTF Open Cloud Standards Incubator as a candidate of a common Cloud interface. vCloud offers an extended concept for the VM, the vApp, that is a software solution containing one or more VMs. The main concepts of the API are the vApp Template, the vApp, the Media (e.g. a CDROM or Floppy Disk) and the network. The protocol used by the API is the REST. The main functions offered by the concepts of the API are presented in Figure 48.

![Figure 48: The vCloud API](image)

2.4.3.1.5. **Oracle Cloud Resource Model API**

The main elements of the Oracle Cloud Resource Model API [67] are the Cloud, the Virtual Data Center, the zone (i.e. logical boundary where the resources may reside), the Service Template (i.e. definition of the deployable service), the Server, the Virtual Network, the volume (i.e. storage),
the Network Interface and the Archive (i.e. a point-in-time representation of a Server). The Cloud does not offer predefined images to instantiate the VMs, but it allows creating a VM manually and capturing a backup snapshot of it. The API uses the REST protocol. The main functions offered by the concepts of the API are presented in Figure 49.

2.4.3.1.6. **ElasticHost API**

The main elements of the ElasticHost [75] API are: the drive, the server, the Virtual Network and the IP Address. The server is a VM working as a server, the driver is a virtual hard disk drive associated to a server. The protocol used by the API is REST. Figure 50 illustrates the main functions offered by API’s concepts.
2.4.3.1.7. **FlexiScale API**

The FlexiScale API [76] supports the server, disk, network, image and firewall concepts. The server runs its own operating system and RAM while it is equipped with at least one disk. The disk is provided by a Storage Area Network (SAN). The FlexiScale API allows the user to create virtual networks and secure them by using a firewall. This API uses the SOAP protocol. Figure 51 presents an overview of the main functions offered by API’s concepts.

![Figure 51: The FlexiScale API](image)

2.4.3.2. **PaaS APIs**

PaaS APIs have increased popularity lately providing higher level functionality. In the following the most representative examples of PaaS APIs are discussed including:

- Amazon PaaS API [78];
- Azure Services Platform API [79];
- Google App Engine API [80];
- Heroku API [81];
- Oracle PaaS Platform API [82];
- VMware API [83];
- Salesforce API [84]; and
- Red Hat API [35].

2.4.3.2.1. **Amazon PaaS API**

Amazon has recently introduced AWS Elastic Beanstalk [78]. Using this, developers can easily create, deploy, and manage applications running on Amazon Web Services Cloud resources. The basic functionality offered by the API is the deployment of the application to the Environment Manager which is based on a set of Configuration and the application Versioning which enables the creation and management of the application’s versions. The main services offered by the API are presented in Figure 52.
2.4.3.2.2. **Azure Services Platform API**

Microsoft’s Azure Services Platform [79] can be used both by applications running in the Cloud and by applications running on local systems. The basic services of Azure API include the Access Controller for handling the user’s access, the Service Bus for exposing services, the Workflow constructor for creating new workflows, the SQL for storage capabilities and the Live for enabling the data access and synchronization. Figure 53 illustrates Azure API main functionality.

2.4.3.2.3. **Google App Engine API**

Google App Engine [80] allows developers to run Web applications on Google's infrastructure. App Engine applications are easy to build, maintain, and scale when traffic grows. The main capabilities offered by the Google’s API include the Mail for sending mails, the URL Fetch for fetching resources, the Messaging for instant messaging, the User handler for users’ authentication, the Datastore for storing objects, the BlobStore for storing large objects, the Task Queue for the background processing, the Image Handler for the manipulation of images and the Memory Cache which enables the usage of the cache memory. Figure 54 illustrates the main functions offered by Google API.
2.4.3.2.4. **Heroku API**

Heroku [81] is a Cloud application platform for building and deploying web applications. It enables developers to concentrate on their applications’ code leaving the management of servers, the deployment, the ongoing operations, and the scaling to be operated by the platform. Its API offers a set of basic services including the Data Base, the Task Manager, the Memory Cache, the Mail, the Bag Tracker, the Search Engine, the Video Encoder, the Security, the SMS Manager and the Document Manager services. Figure 55 illustrates Heroku’s API main functionality.

2.4.3.2.5. **Oracle PaaS Platform API**

The Oracle PaaS Platform [82] is a comprehensive portfolio of products to build an application platform delivered as a public or private Cloud service. The main services offered by its API include the SOA services for the creation of reusable components, the User Interface service that facilitates the users’ interaction, the Business Process Management service that enables the management of the applications, the Database grid service that provide storage capabilities and the Identity Management service that enables the access control.
2.4.3.2.6. **VMware API**

VMware offers vFabric [83] an application Platform for Virtual and Cloud Deployments. vFabric includes the open source Spring Development Framework, as well as a set of services including the Application Server where the applications can be built and run, the Information Manager that enables the exchange of information among the applications, the Application Manager that enables the management of the applications, the Monitoring that monitors the applications and the Enterprise Ready Server that is an extension of the Apache. The services offered are shown in Figure 57.

![Figure 57: The VMWare PaaS](image)

2.4.3.2.7. **Salesforce API**

The main services offered by Salesforce are the User Interface that is responsible for communicating with the user, the Development service that can be used to develop new services, the Database service that facilitates the storage of data, the Logic service that enables the automation of the business processes and finally the Integration service that facilitates the Integration of the applications to diverse environments[84]. The services offered are shown in Figure 58.
2.4.3.2.8. **Red Hat API**

Red Hat PaaS [35] solution provides an open environment that ensures maximum flexibility in development and deployment capitalizing on a set of integrated runtime services and management capabilities that span the entire application lifecycle. Its API consists of Cloud Engine services and PaaS services which are further divided into Application Platform services and Business and Integration services. The Application services consist of several sub-services such as Container, Transaction, Data, Messaging, Web, Cloud-aware clustering and Storage services. On the other hand the Business and Integration services offer integration capabilities by the means of Integration, Business Process, Rules and Presentation/User Interaction services. Lastly, the Cloud Engine facilitates the provisioning and the management of IaaS and PaaS resources. Figure 59 illustrates the main functions offered by Red Hat’ API.
2.4.3.3. **Broker APIs**

These APIs which act as broker API include:

- LibCloud API [77];
- DeltaCloud API [68];
- RightScale [86];
- Enomaly [87];
- OpenStack [88]; and
- OpenNebula API [89].

### 2.4.3.3.1. **LibCloud API**

The LibCloud [77] is a client library that can handle computing resources from diverse Cloud providers. LibCloud is supported by the Apache Software Foundation.

### 2.4.3.3.2. **DeltaCloud API**

DeltaCloud [68] is an API supported by RedHat, which is a part of the Apache Software Foundation Incubator. This API tries to abstract the differences between diverse Clouds offering one entry point for all the Cloud offering (although they may be incompatible). Delta Cloud can handle only computing resources. Its main services are illustrated in Figure 60.

![Figure 60: The DeltaCloud API](image)

### 2.4.3.3.3. **RightScale API**

RightScale’s Cloud Management Platform [86] is a web based Cloud computing management platform which facilitates the deployment and the management of applications spanning multiple Cloud infrastructures – private, public, or hybrid. It delivers complete automation, support for complex deployments, while ensures flexibility, control, and portability. Its main components are shown in Figure 61.
2.4.3.3.4. **Enomaly API**

Enomaly Elastic Computing Platform (ECP) [87] empowers service providers (telcos, hosting providers, and managed service providers) with a complete Cloud in a box platform that enables them to offer revenue-generating Cloud hosting (infrastructure-on-demand, infrastructure-as-a-service, or IaaS) services to their customers. The main components of Enomaly’s API are shown in Figure 62.
2.4.3.3.5. **OpenStack API**

OpenStack [88] is a free, open-source platform that service providers can use to offer infrastructure services similar to Amazon Web Services' EC2 and S3. It has two main parts: a) Nova, originally developed by NASA for its computer processing services, and ii) Swift, the storage service component developed by Rackspace. The OpenStack API can handle compute and storage resources.

2.4.3.3.6. **OpenNebula API**

OpenNebula [89] is an open-source Cloud computing toolkit for managing heterogeneous distributed data center infrastructures. OpenNebula orchestrates storage, network, virtualization, monitoring, and security technologies to deploy multilayer services as virtual machines on distributed infrastructures, combining both data center resources and remote Cloud resources, according to allocation policies. Its main components are illustrated in Figure 64.

![Figure 63: The OpenNebula API [89]](image)

2.4.3.4. **Cloud-Standard Cloud API**

Lastly, the Cloud computing community [49] defines the ideal Cloud API which should provide a Functional and a Management interface [90]. The clients use the Functional interface to reach the functional operations while the management interface deals with the provision and the general management of the Cloud resources. The Cloud API should support a number of functions, such as:

1. Reporting, SLA management, Billing, Metering, Monitoring
2. Security Services
3. Provision, Deploy, Configuration and Control
4. Service Catalogue
5. SLA including elasticity rules, performance, adjacency, compliance, isolation, availability

---


© Cloud4SoA consortium
2.4.3.5. **Elastic Modeling Languages**

A set of modelling languages are introduced in [51]. To date there has not been an end-to-end approach for describing an architecture, from design, through configuration and operations. Therefore, the Elastic Modeling Languages aims to define an end-to-end model-driven approach to IT service design, management and automation which provides great potential to create and maintain the next generation of “Cloud applications”. In particular, the Elastic Modeling Languages express the end-to-end design requirements, control and operational specifications, data centre resources and configurations required to enable automated application deployment and management. They consist of a) the Elastic Computing Modeling Language (ECML), the Elastic Deployment Modeling Language (EDML) and the Elastic Management Modeling Language (EMML) (Figure 65). The ECML is a multi-viewpoint architecture description language, the EDML is a collection of elements for describing the capabilities of IT software and hardware infrastructure and the EMML is a model of configuration items and annotations which describes the context, the state and the dependencies among items.
2.4.4. **Cloud Computing Meta-models**

The following Cloud computing meta-models have been reviewed:

- The Meta-model of Cloud computing components and resources [91]; and
- The simplified Cloud meta-model [92].

2.4.4.1. **Meta-model of Cloud computing components and resources**

A meta-model applicable to any Cloud platform is proposed by [91] (Figure 66). The meta-model represents an abstraction of the different components making up a Cloud and the relations among them.

![Figure 66: Meta-model of Cloud computing components and resources [91]](image)
2.4.4.2. Simplified Cloud meta-model

A simplified meta-model of the Cloud is introduced in [92]. The meta-model is depicted as a UML class diagram (Figure 67) which can be used by the decision making engine to find relevant information about each layer.

![Figure 67: Simplified Cloud meta-model [92]](image)

2.4.5. Discussion

Table 4 shows an overview of the Cloud Service Models presented, explaining what a service can be in the Cloud. The services are separated into four main categories namely the Infrastructure as a Service, the Platform as a Service the Software as a Service and the Service as a Service. The first three categories (i.e. Infrastructure as a Service, the Platform as a Service the Software as a Service) are common in the related literature. Note that at the Cloud Computing stack [57] the Software as a Service is not explicitly defined, but services that belong to this category are included (e.g. Process as a Service and Application as a Service). The Service as a Service category is explicitly defined only in the taxonomy of Cloud Computing services proposed by Intel [55], but the Cloud Computing stack [57] defines services that can be categorized under this category (e.g. Testing as a Service and Integration as a Service). Each of the four main categories is further divided to subcategories. For example the Infrastructure as a Service is divided to Hardware as a Service, Storage as a Service, Computing as a Service etc.

<table>
<thead>
<tr>
<th>Table 4: Cloud Service Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>Hardware as a Service</td>
</tr>
</tbody>
</table>

© Cloud4SoA consortium
<table>
<thead>
<tr>
<th>Service as a Service</th>
<th>✔</th>
<th>✔</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Computing as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Communication as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Information as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Data as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Business Intelligence as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td><strong>Platform as a Service</strong></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Application Development as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Business Analysis Process as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td><strong>Software as a Service</strong></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Process as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Application as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td><strong>Service as a Service</strong></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Billing as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Metadata as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Service Bus as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Security as a Service</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Integration as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Testing as a Service</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Management/Governance as a Service</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
Table 5 presents an overview of the Cloud Resource Models presented, defining the types of resources used in the Cloud. Some resources are grouped together and are presented as one e.g. the network, network interface and IP address are all presented under the resource network. Moreover some models use different names for the same resource e.g. for the Instance many names are used such as Server, Virtual Machine etc.

<table>
<thead>
<tr>
<th>Network</th>
<th>Instance</th>
<th>Storage</th>
<th>Image</th>
<th>Cloud</th>
<th>Virtual data center</th>
<th>Zone</th>
<th>Instance type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Han &amp; Sim [63]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Haase et al. [40]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle [67]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Delta Cloud [68]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>OCCI [69]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon [70]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 6 shows an overview of the Cloud APIs Resource Models. The Amazon API, the Oracle Cloud Resource Model API and the DeltaCloud API are not listed in the table since the corresponding resource models are already presented in Table 5.

<table>
<thead>
<tr>
<th>Network</th>
<th>Instance</th>
<th>Storage</th>
<th>Image</th>
<th>Instance type</th>
<th>Load balancer</th>
<th>Firewall</th>
</tr>
</thead>
<tbody>
<tr>
<td>RackSpace [72]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GoGrid [73]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vCloud [74]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elastic host [75]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlexiScale [76]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LibCloud [77]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloud computing community [90]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Observing Table 5 and Table 6 we can see that all the Cloud Resource Models and the Cloud APIs use similar concepts with similar properties and actions. Hence, common semantics apparently exist. However, the different names and structures used make it difficult to seamlessly move from one Cloud provider to another, as the change of the API may require application refactoring or data transformation. Thus need for a common standardized Cloud API is apparent.
Table 7: Cloud PaaS APIs

<table>
<thead>
<tr>
<th>Application</th>
<th>Environment manager</th>
<th>Store</th>
<th>User Manager</th>
<th>Workflow</th>
<th>Cache</th>
<th>Messaging</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon [78]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Azure [79]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Google App Engine [80]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heroku [81]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle [82]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMware [83]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Salesforce [84]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Red Hat [85]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7 shows an overview of the functionalities offered by the PaaS APIs. It does not present all the concepts that exist in the APIs but only the most commonly used ones. All PaaS APIs share the Application concept. Other popular concepts among the APIs are the Store, the Workflow and the User manager.

Table 8 Cloud Brokers APIs

<table>
<thead>
<tr>
<th>Instance</th>
<th>Network</th>
<th>Storage</th>
<th>Management</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibCloud [77]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaCloud [68]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RightScale [86]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Enomaly [87]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OpenStack [88]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenNebula [89]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Finally, Table 8 presents an overview of the concept and functionalities offered by the Cloud Brokers APIs. On the one hand, some of the APIs offer only broker functionalities (LibCloud, DeltaCloud and OpenStack). On the other hand the RightScale, the Enomaly and the OpenNebula except from the broker functionalities, they offer the capability to manage and monitor the Clouds. An interesting observation is that all the Broker APIs refer to the IaaS layer and none of them manage the PaaS layer.
2.5. Intelligent and Adaptive Front-ends for Cloud Computing

Nowadays, the importance of human-computer interaction principles in designing the user interface of adaptive or “intelligent” systems is being acknowledged by both researchers and developers, especially at enterprise level. The growth in the field of adaptive or intelligent systems has demonstrated the need of investigating the characteristics of the application at service level as well as the design of the interface.

In a SOA, adaptive systems can achieve their main goal, which is improving human-computer interaction, by means of applications. In detail, applications utilize services as fundamental elements to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments, especially at enterprise level [93].

The design of the adaptive interface should meet specific requirements of individual users, such as using personalization of SOA within the context of an intelligent system. On the other hand, the focus of the interface should be the human user instead of the application itself. Therefore, systems should follow a more user-centered design approach than an application-centered.

Section 2.5.1 describes the adaptive and intelligent systems while Section 2.5.2 reviews SOA and mash-ups with focus on Web 2.0 user interfaces.

2.5.1. Adaptive and Intelligent Systems

Literature review indicates that the term “adaptive” is often taken to mean “intelligent” and these two concepts are used as synonyms by researchers in the context of applications of advanced human interactions with the system. Therefore, within this State of the Art review the concepts of adaptive and intelligent will be used interchangeably.

The term of adaptive refers to the systems which can alter aspects of their structure, functionality or interface in order to accommodate the different needs of individuals or groups of users and the changing needs of users over time [94]. In the literature, user-adaptive systems including recommender systems and context-sensitive systems have been suggested as a way to improve human-computer interaction in situations where different users might have different goals, needs and knowledge [95]. Therefore, adaptive systems automatically alter aspects of the system in order to suit the requirements of individuals and groups of users, or more generally of other agents in the system [96].

In the literature, some intelligent user interface adaptations vary from systems targeting user's physical characteristics to systems targeting user preferences in the context of human-machine interaction. For instance, the study of Hagen and Sandnes [97] reports the experiences of universally designed kiosk prototypes based on a multimodal intelligent user interface. This interface was capable of adapting the height of the display of the kiosk with the user's height. However, the focus of the current document is going to be the adaptivity of a system with regard to user preferences and context rather than his physical characteristics.

The interaction between a user and a computer-based system occurs via a user interface. This interface facilitates the communication of a human with a software program or product. There are several approaches to research on intelligent human-computer interaction in the literature, ranging from hardware and software issues of the intelligent interface to speech-input systems. For example, an intelligent system can offer speech recognition software not only to help disabled users (e.g. [98]) but also to control the use of a computer mouse and keyboard (e.g. [99]).
An interface could be considered as intelligent to the degree that it helps users adapt to changing contexts within the field of activity [100]. There are several benefits of these kinds of systems. For instance, intelligence of the computer-based system can ease the burden on users. In the literature, intelligent user interfaces are also proposed as a means to make the system both individualized and personalized by increasing the system’s flexibility and appeal [101].

Research on intelligent user interface also includes the understanding of human intentions. The study of Agah and Tanie [102] proposes the idea that the intentions of the user can be understood by the intelligent system and transformed into implicit commands. Based on this idea, they present a new approach to the design of the intelligent graphical user interfaces. In their study, the users’ movements of the computer mouse to control the cursor on the screen are interpreted as implicit commands by means of the intelligent controller, which was based on multiple software agents, in order to extract the user’s intentions. Then, the mouse cursor is moved by the system according to the intentions of the user which reduced the amount of work required in the interaction with the computer system through the interface.

### 2.5.1.1. Adaptive and adaptable systems

Research on user interface design tends to use the concepts of adaptable and adaptive interchangeably. However, Keeble and Macredie [103] pointed out that using these terms interchangeably may be misleading. When it comes to adaptability or adaptivity, systems where the user is in control of initiation, proposal, selection and production of the adaptation are called adaptable [104]. Adaptable systems refer to customizable systems and allow users to express their preferences to the functionality or the appearance of a user interface. In other words, an adaptable user interface provides tools and allows the user to personalize certain aspects of the interaction while using a system. In contrast to adaptable systems, systems that perform all steps autonomously are called adaptive [104]. In other words, an adaptive or intelligent user interface is one where the appearance, the function or the content of the interface can be changed by the application itself in response to the user’s interactions with it. Therefore, an adaptive or intelligent user interface is broader than the concept of adaptability and changes dynamically in response to its interaction with users [105]. Adaptability and adaptivity can coexist in the same application [104].

### 2.5.1.2. Intelligent agents

The term of intelligent agents offers new insights to the study of intelligent user interfaces [105]. Prior research has presented several definitions of intelligent agents. According to Benyon and Murray [96], intelligent agents are entities which are capable of voluntary and rational action carried out in order to achieve certain goals. A more recent definition of Haynes et al. [106] is “Intelligent agents are software programs designed to act autonomously and adaptively to achieve goals defined by their human developers or runtime users (the latter can be other intelligent agents)” (p.91). Intelligent agents assist users when they work and can perform a specific action as autonomous systems (software or hardware). Therefore, they can sense, act and communicate with other agents, while working on the behalf of other entities [102].

According to Benyon and Murray [96], intelligent interface agents are seen as specialized, knowledge-based systems acting on behalf of the user in some aspect of the interaction. The study of Akoumianakis et al. [105] into intelligent interface agents focuses on both a communication and a collaboration-oriented paradigm for computer-mediated human activities which can decrease human workload and make the overall experience of interaction less stressful and more productive on behalf of the users.
2.5.1.3. **Context sensitivity**

Another concept related to adaptive or intelligent systems is context-sensitivity. In literature the terms context-sensitive, context-based or context-aware systems are used as synonyms and part of intelligent systems. The efforts of making ubiquitous computing applications more attractive and adaptable lead to these kinds of systems. They can be defined as computer systems that use context to provide more relevant functionalities or information to support users while performing their tasks [107]. Context is information used to characterize the situation of an entity, where an entity is a person, place or object that is considered relevant to the interaction between a user and an application. Context-sensitive systems improve the user’s awareness of the task being performed as well as the system adaptation to the user to ease the task execution.

2.5.1.4. **Customization of Intelligent User Interfaces**

Customization or personalization efforts on the Web cover a broad area, ranging from check-box customization, in which portals allow users to select the links they would like on their personal pages, to recommender systems (discussed in the next section) and adaptive Web sites [104]. Customization can be defined as a process that changes the functionality, interface, information content, or distinctiveness of a system to increase its personal relevance to an individual [108]. More specifically, Web customization is performed by applying one or more actions predefined by the interface designer. These actions adapt the information or services provided by a Web site to the knowledge gained from the users’ navigational behavior and individual interests [109].

Regarding these definitions, adaptive hypermedia applications and customization stand close to each other. As already mentioned, adaptable systems refer to customizable systems and allow users to express their preferences to the functionality or appearance of a user interface, while adaptive ones perform this customization on behalf of the user.

Macías and Paternò [110] explore the customization of Web-based applications and address the issue of how to provide intelligent support for customizing Web applications even for non-computer-skilled end-users. In their study they identify the user’s preferences by:

- allowing the user to modify Web pages
- examining the changes the user made to the Web pages with respect to the original pages
- comparing those changes to the changes that could have been made using predefined customization rules
- in case of a match with one or more rules, associate those rules with the user preferences (rule activation)

Their system uses two different kinds of customization rule activations, namely pending and permanent rules. Pending customization rules are identified whenever the system detects a probable user attempt to customize an element or a group of elements while the permanent ones correspond to pending rules that have been identified more than twice within the same context. Therefore, their system enabled to differentiate the occasional user modifications from the repetitive ones.

According to Macías and Paternò, the process of knowledge extraction from the user changes is as follows: when a change to a Web page is made by the user, the system analyses the structure of the page and elements within that structure in order to identify its context. Then, the system checks whether there is an applicable customization rule for that context.

In addition, Macías and Paternò test the effects of both semantic and syntactic changes on user preferences. Semantic changes modify the logical structure of the user interface or the tasks
supported by the interactive application and refer to the transformations of different kinds of interactions, deletions of interactors inside a grouping and/or insertion of interactors inside a grouping. On the other hand, syntactic changes refer to the modifications associated with user preferences and have an effect only on concrete specifications such as changes in text font color preferences, image attribute preferences, background color preferences, button attribute preferences, attribute preferences on links and graphical links. According to their findings, most of the users’ changes were syntactic, rather than semantic. Nevertheless, although semantic changes are less frequent than syntactic ones, they imply deeper modifications of the application. Therefore, they have a deeper impact to end-users when they do not feel satisfied about the semantics of the application.

Customization can be considered from the service provision point of view [111]. Since service providers offer the same kinds of application to many customers, customized applications based on customer preferences and needs are necessary. Service provision can achieve customization in two ways, either through customer specific coding which replaces or complements the components of the application or by specifying the data on which the application operates.

We will discuss further how customization applies to Cloud4SOA in the context of Web 2.0 mash-up techniques.

2.5.1.5. Recommender Systems

Recommender systems help users in searching domains such as e-commerce Web sites and Digital TV that usually provide overwhelming information. Recommenders automatically select and suggest the items that may be appealing to the user considering the preferences defined in his personal profile. An item can be anything in which a user has interest, such as pictures, videos, documents, single files, sequences of files, websites, learning paths, specific tools and applications or people [112]. A broader function of these systems is guiding users in selecting useful or interesting items among a large amount of options in a personalized way [113].

Most recommender systems are based on either collaborative filtering or content-based filtering [113]. This categorization is mostly dependent on the type of algorithm used by the system. Content-based recommenders suggest the items which are useful or interesting to the user in terms of his past preferences. Therefore, they need to create user profiles either by asking the users about their interests explicitly or by a user's given ratings to be able to suggest these interesting items implicitly. However, there are two issues about content-based recommending [112]. The first problem is related to new users because the recommendations of the system need to be developed. The second problem is related to the content-based filtering techniques for profiling the items because they are both time consuming and expensive.

Collaborative filtering constructs groups of like-minded users with whom the target user shares similar interests [114]. Its techniques depend on the items that have been rated previously by other users. The user can rate some items either explicitly such as rating a book on a Web site or implicitly such as visiting a Web site. There are two types of methods for collaborative filtering. The focus of user-based collaborative recommenders is the similarities between the users. System utilizes and analyzes these similarities in order to suggest best rated items by users most similar to the target user. On the other hand, item-based collaborative filtering calculates the similarities between items based on the user's explicit and implicit ratings instead of comparing users.

---

Examples of interactors at the abstract level are Description, Navigator, Text, Single Selection, etc., whereas examples of interactors at the concrete level are Image, Link, Text Field, Radio Button, List, etc.
Nevertheless, in collaborative recommender systems the new user problem still exists with a new additional one: the new item issue. For instance, if an item is rated by a few users, the system will not recommend it.

Blanco-Fernández et al. [115] study the user preferences in recommender systems and suggest that the semantic associations can provide additional knowledge about the user preferences and allow the system to compare them with the available items in a more effective way. Although they use a content-based method that suggests items similar to those the user liked in the past, the recommender system in their study advises items which are semantically associated with user preferences. They also suggest formalizing the domain of the recommender system in a normalized knowledge base in order to extract the semantic associations.

User modeling should be taken into consideration in any context when users will be recommended some services by the platform. User modelling depends on either knowledge-based or behavior-based information in recommender systems [116]. Knowledge-based models utilize the static user models and match users to the closest model. Questionnaires and interviews are often employed to obtain this user knowledge. On the other hand, behavior-based models utilize user behaviors itself as a model and detect the useful patterns of this behavior, e.g. by machine-learning techniques.

The general user profiling approach used by most recommender systems is behavior-based, because they do not require deep engineering such as knowledge-based approaches [117]. In Cloud4SOA behavior-based data can be unobtrusively monitored by analyzing what information the user asks the system to provide and what choices the user makes afterwards. This unobtrusively monitoring technique can lead to content-based recommender systems, for example recommend new Cloud providers that correlate with a user’s preferences and requirements. User preferences are going to be captured by this implicit method.

On the other hand, users will be allowed to rate Cloud services or providers. User feedback can be used for recommending the most popular services as well as to profile the user, which in turn enables collaborative filtering techniques.

In conclusion, since Cloud4SOA aims at semantically describe characteristics of Cloud service provider platforms and applications, a recommender system approach based on semantically annotated content seems a natural choice. Furthermore, allowing users to rate resources (i.e. providers) provide a base for integrating collaborative filtering in the approach.

### 2.5.2. User-centric SOA and Mash-ups

Besides traditional business-to-business SOAs, there is a growing interest for the user-centered SOA. A new trend of Web 2.0 User Interfaces is based on the composition of reusable visual components. Those components are referred in the literature as widgets or gadgets. Widgets can be composed to create new Web services and applications (mash-ups). Widgets and mash-ups are terms associated to Web 2.0 and beyond, as reflected in Nova Spivak’s timeline of the past, present and future of the Web [118] (see Figure 68).
According to W3C\(^51\), widgets are “fully-fledged client-side applications that are authored using Web standards such as HTML5 and packaged for distribution. They are typically downloaded and installed on a client machine or device where they run as stand-alone applications, but they can also be embedded into Web pages and run in a Web browser. Examples range from simple clocks, stock tickers, news casters, games and weather forecasters, to complex applications that pull data from multiple sources to be "mashed-up" and presented to a user in some interesting and useful way”.

Additionally, in the W3C Widgets Landscape\(^52\), W3C remarks that a widget is “an end-user’s conceptualization of an interactive single purpose application for displaying and/or updating local data or data on the Web”.

As said above, a widget can run as a stand-alone application (that is, a desktop application) or can be embedded into a Web document that is rendered in a Web browser. In both cases, widgets need a runtime environment that hosts and instantiates them. Most widgets runtime environments offer functionality similar to a Web browser, rendering the HTML documents produced by the widgets.

Therefore, we can distinguish two kinds of widgets:

- Standalone or desktop widgets, which are small software components hosted by a specific widget environment, sometimes known in the literature as a webtop.
- Web widgets, which are HTML/DHTML/XHTML, CSS and Javascript fragments (documents)\(^53\) included into a Web document.

Desktop widgets run on the host computer as any other application. In contrast, Web widgets are hosted on the server-side and inserted into HTML documents\(^54\) before being sent to the browser client. Examples of desktop widgets are Apple Dashboard widgets\(^55\), Microsoft gadgets\(^56\), KDE

---

\(^{51}\) http://www.w3.org/TR/widgets/#introduction  
\(^{52}\) http://dev.w3.org/2006/waf/widgets-land  
\(^{53}\) Other possible technologies used to embed visual elements into a HTML document, such as Flash, etc. are used.  
\(^{54}\) Usually embedded as an iframe or as HTML elements.  
plasmoids\textsuperscript{57}, Google Desktop gadgets\textsuperscript{58}, Yahoo Widgets\textsuperscript{59}. Examples of Web widgets are: Google Homepage Gadgets\textsuperscript{60} (iGoogle gadgets\textsuperscript{61}), WordPress Widgets\textsuperscript{62}.

Widgets are composable and they can be aggregated by a mash-up assembler, typically using a mash-up editor. A web mash-up is a Web application that combines content and services from different sources to create a completely new service. There is also a rich offer of mash-up tools and editors: IBM Mashup Center\textsuperscript{63}, OpenKapow\textsuperscript{64}, Yahoo pipes\textsuperscript{65}, Dapper\textsuperscript{66}, etc. Web Widgets are usually implemented using the common AJAX frameworks. The technology used is Java (such as Google Web Toolkit\textsuperscript{67}, Java FX\textsuperscript{68}), .NET (such as Microsoft Silverlight\textsuperscript{69}) or Flash (such as Adobe AIR\textsuperscript{70}).

Third party content and services are accessible through public interfaces (API), REST services, RSS feeds, etc. Available mash-ups, Web 2.0 APIs and REST services are catalogued for further reuse in Internet sites such as www.programmableweb.com and webmashup.com.

Lizcano et al.\textsuperscript{119} call this novel approach as user-centric SOA and propose both a generic model of a global user-centric SOA and a novel architecture for enterprise mash-up composite applications. This new approach enables the flexibility of these applications and therefore empowers regular users to create personal applications. Architectures such as the one described by Lizcano allow end-users to develop real composite applications on their own and give an active role to them.

More specifically, Lizcano et al.\textsuperscript{119} propose a top-down software development approach. According to this user-centric application framework, the tool which allows the development of mashupable widgets relays on screen flow resources and semantic Web services stored in a catalogue. The application, namely FAST\textsuperscript{71}, creates the building blocks and widgets while the platform, namely EzWeb\textsuperscript{72}, interconnects these building blocks to compose instant applications. Figure 69 shows an example of a widget mash-up being edited in the FAST editor.

\textsuperscript{57} http://techbase.kde.org/Projects/Plasma/Plasmoids
\textsuperscript{58} http://desktop.google.com/plugins/
\textsuperscript{59} http://widgets.yahoo.com/
\textsuperscript{60} http://code.google.com/apis/gadgets/index.html
\textsuperscript{61} http://www.google.com/ig
\textsuperscript{62} http://widgets.wordpress.com/
\textsuperscript{63} http://www-01.ibm.com/software/info/mashup-center/
\textsuperscript{64} http://kapowsoftware.com/
\textsuperscript{65} http://pipes.yahoo.com/pipes/
\textsuperscript{66} http://open.dapper.net/
\textsuperscript{67} http://code.google.com/webtoolkit/
\textsuperscript{68} http://javafx.com/
\textsuperscript{69} http://www.microsoft.com/silverlight/
\textsuperscript{70} http://www.adobe.com/products/air/
\textsuperscript{71} http://fast-fp7project.morfeo-project.org/
\textsuperscript{72} http://ezweb.morfeo-project.org/
The core building blocks of enterprise 2.0 mash-ups in the study of Lizcano et al. [119] are resources which consist of content, data or application functionality. Using REST architecture, these Web-based resources are addressed by a URI, giving browsers and mobile devices accessibility to these resources.

The process is explained in the following:

1. The user identifies a need or series of needs in the form of both data to be displayed and functionality to be offered.
2. The user searches for a widget that meets his needs or more than one widget that he can put together to create a new one.
3. The user manages his new widget in the dashboard which is supported by an enterprise mash-up platform. This mash-up platform enables widgets to intercommunicate with each other and this leads to a creation of hybrid composite Web application which supports a user’s daily work.
4. Alternatively, users can contribute comments and bugs reports without actually creating new resources in the mash-up platform.

SOA4ALL provides a comprehensive Web 2.0 desktop-like suite of SOA tools for the provisioning and procurement of SOA-based applications [120]. This suite follows a webtop or dashboard approach that aggregates different Web widgets created using the Google Widget Toolkit. SOA4All Studio provides, besides the functional SOA4All tools for service provisioning (composition, annotation, etc), procurement (discovery, consumption, etc) and monitoring, i) the widget dashboard, ii) general purpose widgets (forms, charts, lists, graphs, taxonomy selectors, gauges, etc), iii) management services (authentication/authorization, profiling, auditing, etc) and iv) storage services (repositories, RDF, etc). Both widget and service libraries have been used to create the functional modules (applications) of the SOA4All Studio.
Figure 70 The SOA4All process editor (screenshot)

According to literature, the user-centric SOA should embrace some principles that will lead to the acceptance of services by end-user or consumer [121]. Firstly, an end-user should feel fully empowered, for example he should be fully equipped to set up his own personalized operating environment and achieve his tasks within this environment. Second, an end-user should be able to contribute to the resources used in the application as well as share knowledge about these resources. Third, user-centric SOA should allow an end-user to make community-based collaborations in his network. Therefore, end-users without any specific background knowledge about IT can get involved in the development of SOA.

2.6. Requirements derived from the SotA

As discussed in previous sections, current Cloud platforms vary in terms of services offered, architectures, models (e.g. pricing) as well as in many other characteristics. Building a semantically interoperable Cloud architecture, the Cloud4SOA consortium will take into account the current PaaS systems’ functionalities, characteristics and models.

A typical PaaS solution offers seamless deployment to hosted runtime environments so that a developer is able to deploy a PaaS application with a click. It provides management, monitoring and scaling up and down as minimum functionalities along with billing capabilities for charging [122]. Nevertheless, these are not the only capabilities supported by a PaaS system. As presented below, a number of initiatives attempt to investigate the basic functionality that should be implemented by a PaaS system and, especially, by an interoperable PaaS system.

According to Subray, a PaaS system should offer the following features [123]:

1. **Application Development Technology Framework**: The framework must support the management of the entire application lifecycle of design, development, testing, delivery, change control and update of software. Most widely used technology frameworks are built on J2EE, .NET, PHP.
2. **Ease-of-Use**: A PaaS should offer easy to use tools with pre-built widgets, canned UI components, drag-and-drop tools and support for standard IDEs such as Eclipse, NetBeans, IntelliJ. These also facilitate the rapid Application Development.

3. **Business Process Modelling**: Applications should be built using model-driven methodologies in order to offer the ability to quickly adapt to the changing demands. Specifically, a platform must support workflow capabilities and allow for collaboration as part of the business transactions.

4. **Ubiquitous**: The platform of choice should be accessible and available (at acceptable service levels).

5. **Scalable**: The platform should be able to leverage the elastic capacity from the underlying infrastructure, when needed.

6. **Adaptive**: A best-in-class PaaS should provide ability to develop an application and deliver it on multiple run time platforms besides web like Mobile and Rich Client platforms.

7. **Secure**: The platform should address Cross-site scripting, SQL Injection, Denial of Service and Encryption of traffic ingrained into the application development. Additionally a PaaS should support single sign-in capabilities.

8. **Inclusive**: The platform should provide the ability to include/embed/integrate other applications built on the same platform or others.

9. **Portable**: The platform should be agnostic to the underlying infrastructure and allow companies to move the application to another infrastructure.

10. **On-board Tools**: To easily and quickly migrate data from the legacy on-premise application to the application based on Platform, bulk import and transformation tools are necessary.

In the same context, Dave [124] focuses on the basic features of a PaaS, including:

1. Develop, test, deploy, host and maintain on the same integrated environment.

2. In a completely-realized PaaS, the entire software lifecycle is supported on the same computing environment.

3. User experience without compromise.


5. Built-in integration with web services and databases.

6. Support collaboration throughout the entire software lifecycle.

7. Deep application instrumentation of application and user activity, to help developers understand their applications and effect improvements.

Vambenepe [125] introduces the primary characteristics of a PaaS system from a technical perspective. Among other, he addresses that a PaaS system should offer an application component model that supports deployment/configuration across all PaaS types, explicit interactions/invocations between application components and a subset of platform management interfaces that would be exposed to consumer, along with application management. Furthermore, a PaaS should provide consistent, model-based application management interface across all container types and governance of application services that will be aligned with the container management interfaces. Lastly, debugging, authentication, metering and billing tools should also be included.

Charrington [126] describes the essential characteristics of PaaS systems which fall into four categories:
1. Runtime Framework which executes end-user code according to policies set by the application owner and Cloud provider

2. Abstraction. By means of abstraction, a PaaS Cloud eliminates the complexity of deployment and infrastructure configuration

3. Automation. PaaS environments automate the process of deploying applications to infrastructure, configuring application components, provisioning and configuring

4. Cloud Services. PaaS offerings provide developers with services and APIs that help simplify delivering of elastically scalable, highly available Cloud applications

The essential functions of a PaaS infrastructure are also defined in [34]. The functions can be summarized into: a) it can be shared scalable infrastructure dynamically, b) support components sharing (Figure 71), c) support rapid automatic deployment (Figure 72), d) support self-services (Figure 73). More specifically, a PaaS should provide shared scalable infrastructure dynamically which entails add/remove nodes on-line, automatic re-balance of load, automatic adjustment of node failure, high availability etc. On the other hand, the sharing of components can benefit developers to use embedded components, or use the method of connecting services and configuring instances to develop an application. According to [34] a PaaS platform should support rapidly deployment environments, such as OS, middleware and applications, via virtual modules as well as a number of monitoring and management interfaces to enable application owners to set their own policies, which can make appropriate adjustments according to users' feedback.

Figure 71: The structure of sharing components [34]

Figure 72: Software equipment [34]
The PaaS development stack model, presented in detail in 2.3.2, is based upon six requirements, including:

1. Scalable, reliable, and highly available operational platform deployable to multiple Clouds or between Clouds and on-premise systems.
2. Flexibility in deployment, user interface(s).
3. A highly productive environment using industry-standard and widely available tools, technologies, methodologies, and best practices.
4. Data and applications security.
5. Integration capabilities for applications.
6. Tenancy migration as required.

In the same work, the authors state that the key concept of a perfect PaaS is the construction of a flexible development model that utilizes the existing databases, tools and languages, adapts to the emerging development technologies and applies to multiple Cloud types and multiple Cloud developments.

The purpose of the TeciRes\textsuperscript{73} project is to conduct a technical review of the current landscape within Cloud computing platforms. To this end, Chen et al. have released a technical report [127] illustrating the main technical requirements of Cloud platforms, including:

2. Programming APIs which provides VM image management APIs allowing researchers to upload and deploy research applications or a SDK platform for Cloud research application development.
5. Interoperability, a single self-service and management API to control and manage Cloud runtime environments across multiple Clouds.
6. SLA management.

\textsuperscript{73}http://tecires.ecs.soton.ac.uk/
Besides the aforementioned requirement list, a Cloud platform can additionally support elastic and dynamic scaling in and out, self-service provisioning and management, self-management, automatic scaling, cost effectiveness and multi-tenancy on per-usage basis.

In [128], Gonçalves identifies three main requirements of building platforms by the means of mobile operators, namely a common API, building developer communities and creating a marketplace. The common API will reduce lock-in and increase portability, attracting more end-users to visit the platform without fearing of being locked-in. In addition, the common API benefits developers as they can reuse the application code in several platforms. However, the most significant part of this publication is the increasing interest of authors in engaging both developers and customers building developer communities and presenting developers a direct route to the market through a marketplace that effectively reaches the customer.

In [129], the main capabilities of a SaaS conceptual architecture are presented categorized into five groups, namely presentation, security, application, operations and infrastructure. These capabilities should be also provided by PaaS systems:

1. Application (user profile, metadata execution engine, metadata services, workflow, exception handling, orchestration and data synchronization).
3. Presentation (menu and navigation and reporting).
4. Security (identity and federation, authentication and single sign-on (SSO), authorization and role-based access control, entitlement, encryption and regulatory controls).

In [130], the most common functions to build a standardized SaaS platform are established. The authors classify them into technical and business functions. The technical functions are related to technical issues such as database management, configurable user interface, business workflow and integration. On the other hand, the business functions are divided into Market, Scalability, Development, and Communication functions.

In [45], the authors in an attempt to utilize the Cisco’s reference architecture, implement a use case scenario where the following requirements arise:

5. The end-user can log on to a Cloud portal.
6. The end-user can be registered in a Cloud portal.
7. The end-user can verify/update credentials and information.
8. The end-user can view the provided services.
9. The end-user can select to consume a service.

In [131], Dodani describes how to build a Cloud solution by identifying the core requirements of any Cloud system and implementing them as capabilities provided by a Cloud architecture. The main business and technical requirements derived for each actor are summarized in Table 9.

**Table 9: Core requirements of any Cloud system [131]**

| Provider | • Highly efficient service delivery
| | • Cloud services delivery (application, data and IT resources are provisioned and provided in a standardized way) through the web
| | • A simplified interface/API with well-understood service offerings, pricing and contracts
| | • Flexible pricing model
| | • Service support infrastructure to provide differentiated, well-understood, standardized and high-quality services
| | • Service management and a dynamic infrastructure for scaling up and down
| | • A self-service portal exposing a well-defined set of services in a highly automated fashion

| Developer | • A tooling environment for modeling and assembling service elements
| | • An effective means of managing the service lifecycle.

| Business | • Support for workload offerings
| | • Reporting (on usage, meeting SLAs, licences, etc)
| | • Capabilities of charging (billing, invoices, settlement, etc.)

| Technical | • The portal should be handling licensing and security issues to span different Clouds
| | • Self-service support to manage Clouds using a portal
| | • The portal should facilitate access to the catalogue of services

In the same article, three usage scenarios were presented (a. Create a Cloud service and publish it in the service catalogue, b. Requesting/using the service and c. Managing the Cloud environment) illustrating how the architecture provides the appropriate capabilities to address them. The derived requirements per scenario are listed in Table 10.

**Table 10: Requirements derived from lifecycle usage scenarios [131]**

| 1st Usage Scenario: Create a Cloud service and publish it in the service catalogue | • The service developer is able of creating a service template that will be offered to the service consumers
| | • The service template is published in a service catalogue
| | • Tools are provided to the developer to define and develop the service as well as to develop the images associated with the service
| | • Managing the service offering
| | • Contract management
| | • Offer the service catalogue
| | • Publishing the service
| | • Image lifecycle management
| | • Service automation management (set up the resources to run the service)
2nd Usage Scenario:
Requesting/using the service

- The consumer requests a service from catalogue (Provide information for the request including configuration choices, account information and reservation timeframes)
- Automated provisioning of service (reserve and allocate from resource pool, retrieve and configure VM image, retrieve app. Middleware and configure it, configure and deploy software)
- Service delivery catalogue to save/restore the service instance image
- Service request manager to handle requests

3rd Usage Scenario:
Managing the Cloud environment

- Monitoring IT resources
- Monitoring VMs
- Monitor the entire Cloud environment
- Managing accounts
- Metering usage
- Reporting
- Billing
- SLA compliance
- Managing user groups

In [132], Cisco summaries a set of characteristics that should be reviewed before deciding the PaaS Cloud to which we want to migrate our application. The criteria that should be considered are: i) data portability, which means that the customer must be able to export data from his current service provider’s database in a format that can be migrated to other databases, ii) long-term costs, iii) user management and iv) security. In the same work, SLAs are also referred as a primary criterion. However, in the current state of PaaS systems, SLAs differ across organizations and Cloud providers. Therefore, the creation of a benchmark that would standardize the SLAs provided by different Clouds is important for the uniform interoperability across them. Moreover, when evaluating PaaS, enterprises should also consider platform management and scalability. The PaaS platforms should provide management consoles and tools for monitoring and management of applications that are deployed on them. On the other hand, a PaaS environment might offer dynamic scaling (up or down) as an optional feature.

PAL is a fundamental component of Aneka’s architecture [41]. It manages to hide the heterogeneity of the different operating systems and the underlying hardware behind a common interface. Indeed, PAL exposes a set of functionalities, including uniform and platform independent interface for profiling the hosting platform, uniform access to extended and additional properties of the hosting platform, uniform and platform independent access to remote nodes, uniform and platform independent management interfaces. These are also basic requirements that a PaaS system should address.

Red Hat explores the main features of an open PaaS solution in [35]. Building an open, comprehensive and flexible PaaS system requires the following capabilities:

1. Support existing applications, development paradigms, programming models, and languages, and new ones.
2. Support deployment of existing applications to the Cloud of choice.
3. Enable developers to write an application once and deploy it anywhere allowing applications portability.
4. Bridge multiple deployment environments with a common set of tools and services.
5. Facilitate the development and integration of a broad range of applications so that re-use and share functions provided by existing application.

Besides application portability and common set of tools, an interoperable system should support re-use and functions sharing.

Microsoft outlines four “interoperability elements” which are necessary for achieving interoperability and an open Cloud platform [133]. The four elements are listed below:

1. Portability for customer data.
2. Supporting commonly used standards.
3. Providing easy migration and deployment.
4. Ensuring developer choice on languages, runtimes and tools.

GICTF has recently released the required functionalities for inter-Cloud systems and inter-Cloud interfaces [134]. In particular, the required functionalities for inter-Cloud systems fall into five categories:

1. Matching between service consumer’s quality requirements and SLA.
2. Provisioning.
3. Resource management.
4. Service Setup.
5. Releasing resources.

A number of recommendations for the reduction of risk when adopting Cloud solutions is introduced in [135]. Portability and interoperability are considered part of the risk management and security assurance of any Cloud system. Therefore, architectural considerations that help minimize interoperability effect are essential. Below, a number of considerations for PaaS solutions are provided:

1. Use platform components with a standard syntax, open APIs, and open standards.
2. Understand what tools are available for secure data transfer, backup, and restore.
3. Understand and document application components and modules specific to the PaaS provider, and develop an application architecture with layers of abstraction to minimize access to proprietary modules.
4. Transferring monitoring, logging, auditing and control functions to the new provider.
5. Understand control functions provided by the legacy Cloud provider and how they would translate to the new provider.
6. When migrating to a new platform, understand the impacts on performance and availability of the application, and how these impacts will be measured.
7. Verifying that the migrated services or applications are operating correctly in the new provider.

Lastly, Cohen expresses his thoughts about the necessity for a common, interoperable and open set of Cloud computing standards that will enable Cloud systems to work together in a common way [136]. He identifies a number of key points that can feed the creation of a common Cloud computing reference API or standard. Among them, Cloud resource description, which is the ability to describe resources in a common way, is the most important aspect of the standardization effort. RDF can be used for this purpose. Another standardization aspect is the
ability to bridge different Clouds (Cloud resources) in a secure and efficient way building federations of Clouds.

2.6.1. Discussion

The literature review of related Cloud architectures and models allowed us to identify the main Cloud computing stakeholders as presented in Table 11. We draw upon this analysis and identify the main Cloud4SOA stakeholders, namely Cloud-based application developers and Cloud PaaS providers. The requirements identified in the previous sections are then classified into functional (grouped per identified Cloud4SOA key stakeholder) (Table 12) and non-functional (Table 13).

For the identification and specification of the non-functional requirements the ISO-9126 standard [2] has been used. In the following a brief description of this standard, followed by the different types of non-functional requirements of the standard are presented.

The ISO-9126 standard defines six categories of software qualities (or else non-functional types of requirements). Figure 75 demonstrates the standard and its categories whereas each category is further divided in a number of properties.

Moreover, the non-functional requirements have been augmented with two additional categories, i.e. scalability and availability. Scalability indicates the ability of a system to handle the growing amount of work using additional resources while availability refers to the state of a system to provide its required services.

![Figure 75: ISO-9126 Quality Model [2]](image-url)
# Table 11: Stakeholders per architecture

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Stakeholders</th>
</tr>
</thead>
<tbody>
<tr>
<td>eCloudManager</td>
<td>• eCloudManager Guests</td>
</tr>
<tr>
<td></td>
<td>• eCloudManager Operators</td>
</tr>
<tr>
<td></td>
<td>• eCloudManager Admins</td>
</tr>
<tr>
<td>Aneka [41]</td>
<td>• Developers</td>
</tr>
<tr>
<td>InterCloud [42]</td>
<td>• Cloud Coordinators/Service Producers</td>
</tr>
<tr>
<td></td>
<td>• Cloud Brokers/Service Consumers</td>
</tr>
<tr>
<td></td>
<td>• Cloud Exchange/Market makers</td>
</tr>
<tr>
<td>Fujitsu SaaS-PF [43]</td>
<td>• SaaS user</td>
</tr>
<tr>
<td></td>
<td>• Corporate administrator</td>
</tr>
<tr>
<td></td>
<td>• SaaS service provider</td>
</tr>
<tr>
<td>Open PaaS [35]</td>
<td>• Developers/Architects</td>
</tr>
<tr>
<td>RedHat [44]</td>
<td>• Developers</td>
</tr>
<tr>
<td>Cisco [45]</td>
<td>• End-users</td>
</tr>
<tr>
<td>IBM [46]</td>
<td>• End-users/service consumers</td>
</tr>
<tr>
<td></td>
<td>• Application developers/creators</td>
</tr>
<tr>
<td></td>
<td>• Business process designers/service providers</td>
</tr>
<tr>
<td>Oracle [47]</td>
<td>• Central IT users</td>
</tr>
<tr>
<td></td>
<td>• Application users</td>
</tr>
<tr>
<td></td>
<td>• Application owners</td>
</tr>
<tr>
<td>Cloud Development Stack model [48]</td>
<td>• Developers</td>
</tr>
<tr>
<td>Next Generation Cloud architecture [50]</td>
<td>• Infrastructure Providers</td>
</tr>
<tr>
<td></td>
<td>• Service Providers</td>
</tr>
<tr>
<td></td>
<td>• Service Developers</td>
</tr>
<tr>
<td></td>
<td>• End Users</td>
</tr>
<tr>
<td>Three-partite model [49]</td>
<td>• Developers</td>
</tr>
<tr>
<td>Elastra [51]</td>
<td>• Developers</td>
</tr>
<tr>
<td></td>
<td>• End-users</td>
</tr>
<tr>
<td>Cloud Computing Reference Model [10]</td>
<td>• Cloud providers</td>
</tr>
<tr>
<td></td>
<td>• Consumers</td>
</tr>
<tr>
<td></td>
<td>• Intermediaries</td>
</tr>
<tr>
<td>Cloud Computing Model [16]</td>
<td>• Developers/Consumers</td>
</tr>
<tr>
<td>Adaptive PaaS Architecture [26]</td>
<td>• Developers</td>
</tr>
<tr>
<td>Cloud Deployment Model [51]</td>
<td>• Users</td>
</tr>
<tr>
<td></td>
<td>• Providers</td>
</tr>
<tr>
<td>4CaaSt [53]</td>
<td>• Consumers. Retrieval and consumption of services.</td>
</tr>
<tr>
<td></td>
<td>• Service and application providers. Development and offer of highly demanded services.</td>
</tr>
<tr>
<td></td>
<td>• Service aggregators. Build and offering of service compositions.</td>
</tr>
<tr>
<td>CumuloNimbo [53]</td>
<td>• Software vendors</td>
</tr>
<tr>
<td>Cloud-TM [53]</td>
<td>• Programmers</td>
</tr>
<tr>
<td>mOSAIC [53]</td>
<td>• End – users</td>
</tr>
<tr>
<td></td>
<td>• Cloud application developers</td>
</tr>
</tbody>
</table>
Table 12: Functional Requirements collected from the literature

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Requirement</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud-based application</td>
<td>F1. Transparently migrate data/applications (portability)</td>
<td>[35] [133] [123]</td>
</tr>
<tr>
<td>developer</td>
<td></td>
<td>[132] [40] [48]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[52] [53] [51]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[57] [59] [61]</td>
</tr>
<tr>
<td></td>
<td>F2. Manage resources across multiple Cloud providers</td>
<td>[122] [35] [132]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[134] [127] [125]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[40] [50] [51]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[53] [57] [61]</td>
</tr>
<tr>
<td></td>
<td>F3. Configure (business rules, customizable data model and</td>
<td>[130] [35] [46]</td>
</tr>
<tr>
<td></td>
<td>metadata set)</td>
<td>[16] [47] [48] [44]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[53] [59]</td>
</tr>
<tr>
<td></td>
<td>F4. Monitor execution performance in real-time</td>
<td>[122] [35] [132]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[127] [134] [125]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[40] [41] [43] [35]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[10] [48] [46]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[53] [77]</td>
</tr>
<tr>
<td></td>
<td>F5. Receive (SLA violation, failure) alerts</td>
<td>[129] [48] [10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[45] [58] [61]</td>
</tr>
<tr>
<td></td>
<td>F6. Manage the complete lifecycle of a service</td>
<td>[35] [131] [46]</td>
</tr>
<tr>
<td></td>
<td>(deployment, federation and migration)</td>
<td>[45] [53] [57]</td>
</tr>
</tbody>
</table>

74 4CaaSt, CumuloNimbo, mOSAIC, VISION Cloud, REMICS
75 4CaaSt, mOSAIC, VISION Cloud
76 4CaaSt, VISION Cloud
77 4CaaSt
78 4CaaSt
| F7. | Orchestrate all the involved components | [134] [137] [35] [10] [45] [16] [53] $^{79}$ [58] |
| F8. | Reuse and share functions provided by existing application, enabling applications to work together (integration tools) | [35] [48] [124] [123] [130] [52] [10] [50] [48] [49] [47] [46] [44] [53] $^{80}$ [58] [59] |
| F9. | Support backup and restore | [129] [134] [46] [53] $^{81}$ [59] [61] |
| F10. | Use testing tools | [35] [40] [48] |
| F11. | Use a common API (common set of tools) that supports provisioning/deployment/configuration and control across different Cloud resources | [127] [125] [35] [34] [130] [128] [131] [40] [50] [48] [41] [45] [53] $^{82}$ [54] [59] [61] |
| F12. | Use tools for defining and developing the service as well as to develop the images associated with the service | [131] [41] [43] [40] [48] [50] |
| F13. | Search for services (IaaS, PaaS) held by multiple Cloud systems | [137] [134] [40] [51] [35] [53] $^{83}$ [59] [61] |
| F14. | Access and view the provided services listed in a service catalog | [131] [90] [45] [48] [46] [59] [61] |
| F15. | Enable user profiling (personal workspace) | [129] [59] [119] |
| F16. | Support a developers’ community | [128] [130] [53] $^{84}$ |
| F17. | Support a marketplace (application selling business model, SLA adaptation and support, service billing policy) | [128] [130] [134] [42] [53] $^{85}$ [57] [59] [61] |
| F18. | Transfer monitoring, logging, auditing and control functions to the new provider by the means of commonly defining formats | [135] |
| F19. | Verify that the migrated services or applications are operating correctly in the new provider | [135] |
| F20. | Support self-service provisioning, management and scaling | [34] [127] [131] [40] [51] [46] [47] [53] $^{86}$ [61] |
Table 13: Non-Functional Requirements collected from the literature

<table>
<thead>
<tr>
<th>Requirement type</th>
<th>Requirement</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong>&lt;br&gt;(Security)</td>
<td>NF1. Data and applications security</td>
<td>[48] [123] [124] [127] [132] [10] [50] [49] [46] [51] [45] [43] [41] [53] [57] [61] [59]</td>
</tr>
<tr>
<td>NF2. Encryption</td>
<td></td>
<td>[129] [45] [61]</td>
</tr>
<tr>
<td>NF3. User authentication and single sign-on, authorization and role-based access control, security proof</td>
<td></td>
<td>[130] [125] [134] [129] [51] [43] [10] [59] [61]</td>
</tr>
<tr>
<td>NF4. Secure execution of monetary transactions</td>
<td></td>
<td>[124]</td>
</tr>
<tr>
<td>NF5. Licensing and security issues to span different Cloud platforms</td>
<td></td>
<td>[131]</td>
</tr>
<tr>
<td><strong>Functionality</strong></td>
<td>NF6. Supporting commonly used standards, standard</td>
<td>[131] [133] [48]</td>
</tr>
<tr>
<td>(Interoperability)</td>
<td>syntax, open APIs, widely available tools, technologies, methodologies, and best practices</td>
<td>[130] [134] [135] [16] [51] [53](^90) [57] [58] [59] [54] [61]</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>NF7.</td>
<td>Supporting abstraction (it hides many details of systems infrastructure and application infrastructures from developers and their applications)</td>
<td>[35] [16] [51] [52] [40] [53](^91) [61] [54]</td>
</tr>
<tr>
<td>NF8.</td>
<td>Uniform service description (SLA offering), using standard formats</td>
<td>[137] [134] [53](^92) [57] [58] [61]</td>
</tr>
<tr>
<td>NF9.</td>
<td>SLAs with clear policies and guidelines for maintenance and version management of the platform and policies for version compatibility for APIs between the platform and the application.</td>
<td>[132]</td>
</tr>
<tr>
<td>NF10.</td>
<td>Transparency during interaction with user-adaptive systems</td>
<td>[95]</td>
</tr>
<tr>
<td>NF11.</td>
<td>Technology neutral and loosely coupled widgets while supporting location transparency</td>
<td>[139]</td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF12.</td>
<td>Reliable</td>
<td>[48] [124] [48] [53](^93) [57] [54] [59]</td>
</tr>
<tr>
<td>Usability (Operability)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF13.</td>
<td>Automatic and seamless deployment</td>
<td>[133] [122] [34] [53](^94) [57] [54] [59]</td>
</tr>
<tr>
<td>NF14.</td>
<td>Intercept component-to-component communication.</td>
<td>[125] [137] [54] [59] [61]</td>
</tr>
<tr>
<td>NF15.</td>
<td>The development platform and the development tools are hosted in the Cloud and accessed through a browser</td>
<td>[122] [59]</td>
</tr>
<tr>
<td>NF16.</td>
<td>It provides a presentation interface (menu and navigator, user controls, display and rendering, reporting)</td>
<td>[129] [131] [40] [35] [44] [45]</td>
</tr>
<tr>
<td>Usability (Understandability and Attractiveness)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF17.</td>
<td>Ease-of-Use</td>
<td>[123] [53] [57] [58] [54] [59] [61]</td>
</tr>
<tr>
<td>NF18.</td>
<td>The interface should not be obtrusive</td>
<td>[140]</td>
</tr>
<tr>
<td>NF19.</td>
<td>The interface should be understandable and predictable</td>
<td>[140]</td>
</tr>
<tr>
<td>NF20.</td>
<td>The content and functionality of UI layout will be organized logically.</td>
<td>[140]</td>
</tr>
</tbody>
</table>

\(^90\) 4CaaS, mOSAIC, VISION Cloud, REMICS
\(^91\) 4CaaS, CumuloNimbo, VISION Cloud, REMICS
\(^92\) 4CaaS, mOSAIC
\(^93\) VISION Cloud
\(^94\) 4CaaS, VISION Cloud

© Cloud4SoA consortium
<table>
<thead>
<tr>
<th>NF21.</th>
<th>The intelligent user interface should use graphical elements instead of using text-based linear lists because a graphical user interface can increase the level of user satisfaction as well as the level of the communication of the adaptive system with its users</th>
<th>[141]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Efficiency</strong></td>
<td>NF22.</td>
<td>It can be accessed by multiple users at the same time (multi-tenant)</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>NF23.</td>
<td>Scalable infrastructure provisioning as needed</td>
</tr>
<tr>
<td></td>
<td>NF24.</td>
<td>Scalable deployment to multiple Cloud platforms and between Cloud systems and on-premise systems</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>NF25.</td>
<td>It should be accessible and available (at acceptable service levels).</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>NF26.</td>
<td>Ensuring developer choice on languages, runtimes and tools</td>
</tr>
</tbody>
</table>

95 mOSAIC, VISION Cloud
96 4CaaSt, CumuloNimbo, VISION Cloud
97 4CaaSt, CumuloNimbo, VISION Cloud
98 mOSAIC, VISION Cloud

© Cloud4SoA consortium
3. **Cloud4SOA Requirements**

This section aims to introduce the Cloud4SOA main stakeholders and usage scenarios. Section 3.1 documents two usage scenarios of the Cloud4SOA platform that focus on the different functionalities of the Cloud4SOA platform as well as on the benefits that are derived from joining a network with interoperable PaaS providers. Last, section 3.2 summarizes the requirements elicited from the usage scenarios.

### 3.1. Cloud4SOA Usage Scenarios

This section documents the two usage scenarios (US) of Cloud4SOA, namely deploying a service-based application on the Cloud4SOA platform and migration to/deployment on a different Cloud PaaS Provider. The usage scenarios were designed collaboratively by the Cloud4SOA consortium during the first four months of the project. They focus on the different functionalities that the Cloud4SOA platform should offer to address specific requirements of the trial partners, i.e. FIT, PTIN and RomTelecom, and cloudControl, as well as on the expected benefits of the Cloud4SOA solution.

#### 3.1.1. US1: Deploying a service-based application on the Cloud4SOA platform

In this scenario, Fraunhofer FIT, a large German research institute that has developed BSCW, decides to move part of BSCW from their legacy information system to the Cloud.

FIT identifies a new business opportunity by exposing BSCW to their users in an ASP fashion adopting a pay-per-use price model. FIT wants to create business, increasing its market or explore new models, while limiting possible IT risks.

The advantages of a Cloud-enabled BSCW solution would include:

- Better and faster reaction on dynamic use and interaction patterns of BSCW;
- Dynamic adoption to data storage requirements;
- Scalability of BSCW;
- Interoperability of different BSCW installations on a file store and back-end layer; and
- Potential interoperability between BSCW and other Cloud-enabled applications on a data and service level, thus enabling the end-user creation of a seamless cooperation environment.

Hence, Wolfgang, director of the Institute, consults FIT’s technicians and comes up with a list of requirements, regarding QoS, price etc. that the candidate Cloud provider will have to satisfy. Currently, Wolfgang has to perform a manual search in order to find the Cloud providers and then contact each one of them separately asking them whether they can provide FIT with the desired QoS, price etc. levels.

Thanassis, a friend of Wolfgang, tells him about the new platform that has been recently launched by the Cloud4SOA project.

The Cloud4SOA platform provides access to a large number of European small-medium Cloud providers that join forces and compete with big Cloud vendors by interoperating through the Cloud and removing the vendor lock-in issue. The platform promises to find the best match...
depending on the requirements of the client. Thanassis encourages Wolfgang to visit the Cloud4SOA platform available at www.Cloud4soa.eu/platform and try it out.

Wolfgang visits the platform and is impressed by the intelligent adaptive user interface. He selects the “Deploy your application...” choice and the user interface is asking him to fill in the requirements of his application, i.e. the semantic profile of the Web archive that contains the application. Wolfgang is experienced enough to understand that the Cloud4SOA user interface capitalizes on robust Cloud service and resource models. He provides the semantic profile and after a few seconds the platform negotiates and recommends the top 5 Cloud providers that are able to serve his needs. Wolfgang notices that cloudControl, a new promising PaaS provider, is among them. He selects cloudControl and uses the deploy option provided by the Cloud4SOA platform. An intuitive wizard is initiated that allows him to upload the Web archive of his application. His work is over. The Cloud4SOA platform contacts the Cloud4SOA platform which takes care and deploys the Web archive on cloudControl’s platform. The Cloud4SOA platform also allows Wolfgang to manage the Cloud-based BSCW lifecycle, e.g. start and/or stop the service.

Wolfgang is very satisfied that in just a few minutes he managed to find the Cloud provider that will host the Cloud-based offering of BSCW and deploy it too.

3.1.2. US2: Migration to/Deployment on a different Cloud PaaS Provider

FIT have moved BSCW from their legacy information system to the Cloud using the Cloud4SOA platform (US1).

Through the Cloud4SOA widgets FIT monitors the different modules, retrieves statistics about them and takes recovery actions when SLA violation is raised. Lately FIT is not satisfied with the performance of their Cloud PaaS provider, cloudControl, and Wolfgang, director of the Institute, has noticed that the SLAs that they have commonly agreed are breached by the Cloud provider.

Hence, Wolfgang decides to replace them. Wolfgang visits again the Cloud4SOA platform, logs onto FIT’s profile and searches for a PaaS offering that will satisfy the semantic profile of BSCW and will offer better performance at a lower price than what FIT has been paying cloudControl. Wolfgang already knows that all Cloud4SOA-enabled PaaS vendors that use the Cloud4SOA API are interoperable, hence guaranteeing seamless data and application migration.

He selects the “Migrate my application...” option. A wizard is initiated. Wolfgang is surprised to see that the wizard has already loaded the semantic profile of his application, the semantic profile of cloudControl and their SLA. He reviews the stored information and makes any changes necessary (also re-negotiates the SLA with new requirements). The Cloud4SOA platform queries the overlay network of the Cloud4SOA-enabled PaaS providers. The Cloud4SOA platform match makes, negotiates and recommends to Wolfgang the top 5 Cloud providers that meet his application’s requirements. Wolfgang can also consult the available user-generated comments and ratings per PaaS provider. Wolfgang notices that SemanticCloud2.0, a new promising PaaS provider, is among the vendors that match his requirements. He selects SemanticCloud2.0 and uses the “Migration” option provided by the user interface. A wizard is initiated that downloads a Web archive with the data and the applications from cloudControl. Automatically, the Web archive is relocated and deployed on SemanticCloud2.0 platform.

After a while, FIT decides to extend their Cloud-based BSCW with additional services. However, while trying to deploy one of these new services, the Cloud4SOA platform’s widgets inform Wolfgang that SemanticCloud2.0 cannot offer the additional processing power required. In order
to solve this problem, Wolfgang visits the Cloud4SOA platform to search for an interoperable Cloud PaaS provider that can offer him the processing power required for BSCW’s new service.

He selects the “Find additional resources” choice and automatically the semantic profile of his application and the SLA that he has agreed with SemanticCloud2.0 are displayed enabling him to configure them. The Cloud4SOA platform performs the same search that was described in the “Move my application” case. Wolfgang notices that CloudScale, a reliable small-medium Cloud PaaS provider is available. He selects the “Deploy” option and a wizard is initiated that allows him to upload the Web archive of his application to the new PaaS provider.

Wolfgang is very satisfied that he managed to meet his applications needs without using additional software, changing application setting or putting his data at risk.

### 3.2. Requirements derived from the Usage Scenarios

In this section we present the list of requirements that derived from the aforementioned usage scenarios. Table 14 contains the functional requirements (grouped per identified Cloud4SOA key actor, i.e. application developer and Cloud platform provider), while Table 15 the non-functional ones.

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Requirement</th>
<th>US</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud-based application developer</td>
<td>F30. Register in the Cloud4SOA platform</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F31. Enable user profiling (personal workspace)</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F32. Edit/update profile</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F33. Create the application’ s semantic profile</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F34. View the provided services listed in a service catalog</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F35. Search for a Cloud platform provider using specific criteria</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F36. View the recommended PaaS offerings</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F37. Establish agreements with Cloud4SOA enabled providers (when registered)</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F38. Deploy (part of) application on a Cloud platform</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F39. Acquire additional resources upon request</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>F40. Monitor, manage and configure an application’s lifecycle</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F41. View details on the infrastructure and the services provided by each Cloud platform provider</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F42. Rate PaaS providers and the service offered by them (in terms of quality, reliability etc.)</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F43. View the user-generated comments and ratings for a specific PaaS provider</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>F44. Migrate an application from one Cloud platform to another.</td>
<td>2</td>
</tr>
<tr>
<td>Requirement</td>
<td>Description</td>
<td>References</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>F45.</td>
<td>Receive notification when an SLA violation is raised</td>
<td>1, 2</td>
</tr>
<tr>
<td>F46.</td>
<td>Get recommendations in selecting a Cloud provider based on a hybrid recommender system approach</td>
<td>1, 2</td>
</tr>
<tr>
<td>F47.</td>
<td>Register in the Cloud4SOA platform (corporate workspace)</td>
<td>1, 2</td>
</tr>
<tr>
<td>F48.</td>
<td>Edit/update profile</td>
<td>1, 2</td>
</tr>
<tr>
<td>F49.</td>
<td>Map their API to the Cloud4SOA API</td>
<td>1, 2</td>
</tr>
<tr>
<td>F50.</td>
<td>Define their generic pricing model and the associated performance characteristics (semantic profile of his platform) and publish them in the Cloud4SOA platform</td>
<td>1, 2</td>
</tr>
<tr>
<td>F51.</td>
<td>Establish agreements/form alliances/collaborate with other providers in the Cloud4SOA marketplace platform</td>
<td>1, 2</td>
</tr>
<tr>
<td>F52.</td>
<td>Be rated negatively when violating a negotiated SLA</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

### Table 15: Non-functional requirements collected from the US

<table>
<thead>
<tr>
<th>Requirement type</th>
<th>Requirement</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality (Security)</strong></td>
<td><strong>NF27.</strong> User level authentication</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF28.</strong> Data security/accuracy and Data privacy</td>
<td>1, 2</td>
</tr>
<tr>
<td><strong>Functionality (Interoperability)</strong></td>
<td><strong>NF29.</strong> Semantic interoperability between Cloud platform providers</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF30.</strong> Supporting commonly used standards, standard syntax, open APIs, widely available tools, technologies, methodologies, and best practices.</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF31.</strong> Use common service descriptions (Provision of common terminology e.g. semantics, abstracting providers’ heterogeneity)</td>
<td>1, 2</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td><strong>NF32.</strong> No data loss</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF33.</strong> Error prevention</td>
<td>1, 2</td>
</tr>
<tr>
<td><strong>Usability (Operability)</strong></td>
<td><strong>NF34.</strong> It provides a presentation interface (widgetized services)</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF35.</strong> The user interface provides support for all the modules that requires interaction with the user (searching, monitoring and governance)</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF36.</strong> Usability principles for intelligent interfaces need to be defined.</td>
<td>1, 2</td>
</tr>
<tr>
<td><strong>Usability (Understandability and Attractiveness)</strong></td>
<td><strong>NF37.</strong> The user interface should adapt to the user’s profile</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF38.</strong> There should be contextual help and documentation available</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>NF39.</strong> The user interface should use graphical elements instead of using text-based ones</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

© Cloud4SoA consortium
| NF40. | The user interface should be easy to use and intuitive. | 1, 2 |
| NF41. | The platform should be configurable by e.g. allowing the removal/addition of functionalities/technologies (version upgrades) | 1, 2 |
| NF42. | Scalable infrastructure provisioning as needed | 1, 2 |
| NF43. | Available at any time | 1, 2 |
| NF44. | It should be accessible and available (at acceptable service levels). | 1, 2 |
4. **Cloud4SOA Reference Framework Requirements**

This section reports the results of the Cloud4SOA requirements prioritization exercise. The functional requirements identified in sections 2.6 and 3.2 have been integrated in one single list which is presented in Table 16.

Likewise, the two lists of non-functional requirements have been merged in the list presented in Table 17.

During the integration process, some of the requirements of the two sections 2.6 and 3.2 have been eliminated as they are beyond the scope of the Cloud4SOA project and do not contribute to the implementation of the project’s vision. These requirements are: F7, F8, F9, F10, F17, F20 and F29 in Table 12; NF4 and NF14 in Table 13.

Additionally, the description of some requirements has been adapted in the case where a requirement coming from the State of the Art and a requirement coming from the usage scenarios had to be integrated.

The prioritization of the requirements was carried out according to the following three levels:

- Top priority was given to all the requirements that originated both in the usage scenarios and the literature review.
- Medium priority was assigned to all the requirements coming only from the usage scenarios.
- Low priority was given to all the requirements that derived only from the literature review.

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Requirement</th>
<th>Reference</th>
<th>US</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud-based application developer</td>
<td>F1. Enable user profiling (personal workspace)</td>
<td>[129] [59] [119]</td>
<td>1,2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2. Be able to search for available PaaS/IaaS offerings using specific criteria</td>
<td>[137] [134] [40] [51] [35] [53] [59] [61]</td>
<td>1,2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F3. Access and view a catalogue of the PaaS offerings registered in the Cloud4SOA platform</td>
<td>[131] [90] [45] [48] [46] [59] [61]</td>
<td>1,2</td>
<td>Top</td>
</tr>
<tr>
<td></td>
<td>F4. Acquire recommendations on Cloud4SOA-enabled platforms</td>
<td>[112]</td>
<td>1, 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F5. Monitor, manage and configure a Cloud-based application’s lifecycle (deployment,</td>
<td>[35] [131] [46] [45] [53] [108] [57]</td>
<td>1, 2</td>
<td></td>
</tr>
</tbody>
</table>

99 mOSAIC

© Cloud4SoA consortium
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>federation and migration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F6.</td>
<td>View details on the infrastructure and the services provided by each Cloud4SOA-enabled platform</td>
<td>[122] [35] [132] [134] [127] [125] [40] [50] [51] [53] [57] [61]</td>
<td>1, 2</td>
</tr>
<tr>
<td>F7.</td>
<td>Be able to migrate an application from one Cloud4SOA-enabled platform to another</td>
<td>[35] [133] [123] [132] [40] [48] [52] [53] [57] [58] [59] [61]</td>
<td>2</td>
</tr>
<tr>
<td>F8.</td>
<td>Receive notification when an SLA violation is raised</td>
<td>[129] [48] [10] [45] [58] [61]</td>
<td>1, 2</td>
</tr>
<tr>
<td>F9.</td>
<td>Register in the Cloud4SOA platform</td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>F10.</td>
<td>Edit/update profile</td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>F11.</td>
<td>Create the semantic profile of a Cloud-based application</td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>F12.</td>
<td>Establish agreements with Cloud4SOA-enabled providers (when registered)</td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>F13.</td>
<td>Deploy (part of) application on a Cloud4SOA-enabled platform</td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>F14.</td>
<td>Rate PaaS providers and the service offered by them (in terms of quality, reliability etc.)</td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>F15.</td>
<td>View the user-generated comments and ratings for a specific PaaS provider</td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>F16.</td>
<td>Acquire additional resources upon request</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>F17.</td>
<td>Configure the Cloud4SOA platform (business rules, customizable data model and metadata set)</td>
<td>[130] [35] [46] [16] [47] [48] [44] [53] [59]</td>
<td>103</td>
</tr>
<tr>
<td>F18.</td>
<td>Use tools for defining and developing the service-based applications and for developing the images associated with them</td>
<td>[131] [41] [43] [40] [48] [50]</td>
<td>104</td>
</tr>
<tr>
<td>F19.</td>
<td>Monitor execution performance in real-time</td>
<td>[122] [35] [132] [127] [134] [125] [40] [41] [43] [35] [10] [48] [46] [53]</td>
<td>104</td>
</tr>
<tr>
<td>F20.</td>
<td>Support a developers’ community</td>
<td>[128] [130] [53]</td>
<td>105</td>
</tr>
</tbody>
</table>

100 4CaaS
t101 4CaaS, mOSAIC, VISION Cloud
t102 4CaaS, CumuloNimbo, mOSAIC, VISION Cloud, REMICS
t103 4CaaS, VISION Cloud
t104 4CaaS
t105 4CaaS

© Cloud4SoA consortium
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Reference</th>
<th>US</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>F21.</td>
<td>Transfer monitoring, logging, auditing and control functions to the new provider by the means of commonly defining formats</td>
<td>[135]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F22.</td>
<td>Verify that the migrated Cloud-based application is operating correctly on the new PaaS offering</td>
<td>[135]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F23.</td>
<td>Ability to determine the privacy level of their data (user profile, personalized environment)</td>
<td>[138] [130] [53] [129] [10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F24.</td>
<td>Define generic pricing model and the associated performance characteristics (semantic profile of his platform) and publish them</td>
<td>[131] [132] [35] [40] [48]</td>
<td>1, 2</td>
<td>Top</td>
</tr>
<tr>
<td>F25.</td>
<td>Establish agreements/form alliances/collaborate with other Cloud4SOA-enabled providers</td>
<td>[35] [124] [42] [53]</td>
<td>1, 2</td>
<td></td>
</tr>
<tr>
<td>F26.</td>
<td>Register in the Cloud4SOA platform (corporate workspace)</td>
<td>1, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F27.</td>
<td>Edit/update profile</td>
<td>1, 2</td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>F28.</td>
<td>Map their platform’s API to the Cloud4SOA API</td>
<td>1, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F29.</td>
<td>Be rated negatively when violating a negotiated SLA</td>
<td>1, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F30.</td>
<td>Monitor the status of the resources (dead/alive) and application components, services, and infrastructure to detect failures.</td>
<td>[129] [131] [134] [61]</td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>F31.</td>
<td>Manage the access to resources/services</td>
<td>[35] [131] [53] [107] [59] [61]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F32.</td>
<td>Manage the SLA contracts</td>
<td>[131] [58]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F33.</td>
<td>Automatic update of a Cloud system’s SLA as a result of a change in the provider’s policy after SLA matching.</td>
<td>[134] [57] [58]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 17: Integrated list of Non-Functional Requirements

<table>
<thead>
<tr>
<th>Requirement type</th>
<th>Requirement</th>
<th>Reference</th>
<th>US</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>NF1.</td>
<td>Data and applications security</td>
<td>[48] [123] [124] [127] [132] [10] [50]</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

106 VISION Cloud, CONTRAIL
107 4CaaS, mOSAIC, VISION Cloud
<table>
<thead>
<tr>
<th>Functionality (Interoperability)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NF2.</strong> User authentication and single sign-on, authorization and role-based access control, security proof</td>
</tr>
<tr>
<td>Medium</td>
</tr>
<tr>
<td><strong>NF3.</strong> Encryption</td>
</tr>
<tr>
<td>[129] [45] [61]</td>
</tr>
<tr>
<td><strong>NF4.</strong> Licensing and security issues to span different Cloud platforms</td>
</tr>
<tr>
<td>[131]</td>
</tr>
<tr>
<td><strong>NF5.</strong> Use common service descriptions (common terminology e.g. semantics, abstracting providers’ heterogeneity)</td>
</tr>
<tr>
<td>[137] [134] [53] [57] [58] [61]</td>
</tr>
<tr>
<td><strong>NF6.</strong> Supporting commonly used standards, standard syntax, open APIs, widely available tools, technologies, methodologies, and best practices.</td>
</tr>
<tr>
<td>[131, 133] [48] [130] [134] [135] [16] [51] [53] [57] [58] [59] [54] [61]</td>
</tr>
<tr>
<td><strong>NF7.</strong> Use a common API (common set of tools) that supports provisioning/deployment/configuration and control across different Cloud resources</td>
</tr>
<tr>
<td>[127] [125] [35] [34] [130] [128] [131] [40] [50] [48] [41] [45] [53] [54] [59] [61]</td>
</tr>
<tr>
<td><strong>NF8.</strong> Semantic interoperability between Cloud4SOA-enabled platforms</td>
</tr>
<tr>
<td>1, 2 Medium</td>
</tr>
<tr>
<td><strong>NF9.</strong> Supporting abstraction (hide details of systems infrastructure and application infrastructure from developers and their applications)</td>
</tr>
<tr>
<td>[35] [16] [51] [52] [40] [53] [61] [54]</td>
</tr>
<tr>
<td><strong>NF10.</strong> Support SLAs with clear policies and guidelines for maintenance and version management of the platform and policies for version compatibility for APIs between the platform and the application.</td>
</tr>
<tr>
<td>[132]</td>
</tr>
<tr>
<td><strong>NF11.</strong> Support transparency during interaction with user-adaptive systems</td>
</tr>
<tr>
<td>[95]</td>
</tr>
</tbody>
</table>

108 VISION Cloud
109 4CaaSt, mOSAIC
110 4CaaSt, mOSAIC, VISION Cloud, REMICS
111 4CaaSt, mOSAIC
112 4CaaSt, CumuloNimbo, VISION Cloud, REMICS

© Cloud4SoA consortium  page 116 of 131
| NF12. | Support technology neutral and loosely coupled widgets while supporting location transparency | [139] |
| NF13. | No data loss | 1, 2 | Medium |
| NF14. | Error prevention | 1, 2 |
| NF15. | Reliable | [48] [124] [48] [53] [57] [54] [59] | Low |
| NF16. | It provides a presentation interface (menu and navigator, user controls, display and rendering, reporting) | [129] [131] [40] [35] [44] [45] | 1, 2 | Top |
| NF17. | The user interface provides support for all the modules that requires interaction with the user (searching, monitoring and governance) | 1, 2 | Medium |
| NF18. | Usability principles for intelligent interfaces need to be defined. | 1, 2 |
| NF19. | Automatic and seamless deployment | [133] [122] [34] [53] [57] [54] [59] | Low |
| NF20. | The development platform and the development tools are hosted in the Cloud and accessed through a browser | [122] [59] |
| NF21. | The intelligent user interface should use graphical elements instead of using text-based linear lists because a graphical user interface can increase the level of user satisfaction as well as the level of the communication of the adaptive system with its users | [141] | 1, 2 | High |
| NF22. | The interface should be understandable and intuitive | [140] | 1, 2 |
| NF23. | Ease-of-Use | [123] [53] [57] [58] [54] [59] [61] | 1, 2 |
| NF24. | The user interface should adapt to the user’s profile | 1, 2 | Medium |
| NF25. | There should be contextual help and documentation available | 1, 2 |

113 VISION Cloud
114 4CaaSt, VISION Cloud

© Cloud4SoA consortium
<table>
<thead>
<tr>
<th>NF</th>
<th>Description</th>
<th>References</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF26</td>
<td>The content and functionality of UI layout will be organized logically.</td>
<td>[140]</td>
<td>Low</td>
</tr>
<tr>
<td>NF27</td>
<td>The interface should not be obtrusive</td>
<td>[140]</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td><strong>Maintainability (Changeability)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF28</td>
<td>The platform should be configurable by e.g. allowing the removal/addition of functionalities/technologies (version upgrades)</td>
<td>1, 2</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td><strong>Efficiency</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF29</td>
<td>It can be accessed by multiple users at the same time (multi-tenant)</td>
<td>[131] [48] [26] [53] [59]</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td><strong>Scalability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF30</td>
<td>Scalable infrastructure provisioning as needed</td>
<td>[124] [132] [125] [127] [130] [35] [123] [48] [42] [53] [57] [58] [54] [59]</td>
<td>Low</td>
</tr>
<tr>
<td>NF31</td>
<td>Scalable deployment to multiple Cloud platforms and between Cloud systems and on-premise systems</td>
<td>[48] [134] [26]</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Availability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF32</td>
<td>It should be accessible and available (at acceptable service levels).</td>
<td>[48] [123] [46] [53] [57] [58] [54] [59] [61]</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td><strong>Other</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF33</td>
<td>Ensuring developer choice on languages, runtimes and tools</td>
<td>[133] [35] [48] [53] [59]</td>
<td>Low</td>
</tr>
</tbody>
</table>

115 mOSAIC, VISION Cloud  
116 4CaaS, CumuloNimbo, VISION Cloud  
117 4CaaS, CumuloNimbo, VISION Cloud  
118 mOSAIC, VISION Cloud
5. Conclusions

The present document is the first deliverable of the first work package in the context of the Cloud4SOA project. The objective of this document was to gather the requirements that will be later on addressed by Cloud4SOA. To succeed this, a first analysis of the State of the Art was necessary, focusing on (i) Cloud computing interoperability, especially on PaaS layer, (ii) Cloud computing architectures which aim to resolve interoperability issues, (iii) semantic service and resource models, meta-models and Cloud APIs and, (iv) intelligent and adaptive interfaces for Cloud computing. After reviewing the State of the Art, a first list of requirements was created. This list was further enhanced with more practical requirements derived from usage scenarios that presented examples of various functionalities needed to be supported by the Cloud4SOA platform.

The functional and non-functional requirements derived from the literature review and the usage scenarios were classified into three levels of priority:

- Top priority was given to all the requirements that originated both in the usage scenarios and the literature review.
- Medium priority was assigned to all the requirements coming only from the usage scenarios.
- Low priority was given to all the requirements that derived only from the literature review.

Table 18 shows the number of functional and non-functional requirements placed at each of the three levels of priority.

<table>
<thead>
<tr>
<th>Functional requirements</th>
<th>Non-functional requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top priority</td>
<td>10</td>
</tr>
<tr>
<td>Medium priority</td>
<td>12</td>
</tr>
<tr>
<td>Low priority</td>
<td>11</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>33</strong></td>
</tr>
</tbody>
</table>

The work reported in this deliverable will be used as the foundation for further work in the project. In particular, the top and medium priority requirements are the ones that will be addressed first in the context of the Cloud4SOA Cloud Semantic Interoperability Framework (D1.2) and the Cloud4SOA Reference Architecture specified (D1.3).
References


© Cloud4SoA consortium


© Cloud4SoA consortium


Annex A. Full license

This is a public deliverable that is provided to the community under the license Attribution-NoDerivs 2.5 defined by creative commons http://www.creativecommons.org

This license allows you to

• to copy, distribute, display, and perform the work
• to make commercial use of the work

Under the following conditions:

![Attribution] Attribution. You must attribute the work by indicating that this work originated from the Cloud4SoA project and has been partially funded by the European Commission under contract number 216759

![No Derivative Works] No Derivative Works. You may not alter, transform, or build upon this work without explicit permission of the consortium

• For any reuse or distribution, you must make clear to others the license terms of this work.
• Any of these conditions can be waived if you get permission from the copyright holder.

This is a human-readable summary of the Legal Code below:

A.1. License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED. BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

A.1.1. Definitions

a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purposes of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.

c. "Licensor" means all partners of the Cloud4SoA consortium that have participated in the production of this text.

d. "Original Author" means the individual or entity who created the Work.

e. "Work" means the copyrightable work of authorship offered under the terms of this License.
f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

A.1.2. Fair Use Rights

Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

A.1.3. License Grant

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.
c. For the avoidance of doubt, where the work is a musical composition:
d. Performance Royalties Under Blanket Licenses. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
e. Mechanical Rights and Statutory Royalties. Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).
f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved.

A.1.4. Restrictions

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimers of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by clause 4(b), as requested.
b. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or Collective Works, You must keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution in Licensor's copyright notice, terms of service or by
other reasonable means, the name of such party or parties; the title of the Work if supplied; and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

A.1.5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

A.1.6. Limitation on Liability

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

A.1.7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

A.1.8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.