Expressing Quality of Service and Protection Using Federation-Level Service Level Agreement

Lorenzo Blasi[1], Jens Jensen[2], and Wolfgang Ziegler[3]

[1] Hewlett-Packard, Italy
[2] STFC, UK
[3] Fraunhofer, Germany

Abstract. Frameworks for service level agreements (SLAs) have been developed to allow services to discover and negotiate SLAs dynamically, without direct human intervention. We give a description of the experiences of two projects which are building on existing work in SLAs: the main advantages being to obtain better overall services (including pricing) for the consumer, and the SLAs are useful as a component of, or extension to, cloud federations. We also argue that the "Quality of Protection" is an important part of SLAs.

1 Background

Traditionally, Service Level Agreements (SLAs) are negotiated between service providers and the customer. The SLA typically describes the terms and nature of the service, as well as specific parameters of the service, perhaps including the cost, target uptime (if it's an online service), level of support, etc. The aim is to have each party agree on the type and level of service provided. Traditionally, SLAs are negotiated between *people* who represent the provider and consumer – in particular, they must have the *authority* to negotiate an agreement.

The availability of online services, and cloud services in particular, changed this scenario. Now the consumer is a person looking for a service, and the provider publishes terms and conditions (T&C), cost, target availability on a web page; and users may need to accept T&C prior to using the service. For a typical cloud service, the customer's needs are small compared to the provider's capacity, and there is no need to enter into negotiations, nor would the provider be interested in negotiating with a large number of small users. When "shopping around" for services, the user can either "take it or leave it'.' For a cloud service, the provider is an automated agent which can deal fully with setting up the new customer. All customers are then treated the same, leading to an economy of scale.

The logical extension is: what if both the customer and the provider are represented by agents? And what if they can not just sign an agreement, but also negotiate terms? In this extended model the human outlines a policy for service delivery by expressing requirements, and leaves it to the agent to look for service providers that match.

The promise of this approach is obvious: now the end user is free to consume services, perhaps simultaneously from multiple providers (assuming of course

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 146–156, 2014.

they are interchangeable). If a provider drops out, or another becomes cheaper, the agent should negotiate new agreements, and services will eventually move to the new provider.

Moreover, brokering and monitoring services from multiple providers can be a means of obtaining higher Quality of Service (QoS), at least availability and perhaps bandwidth, as in the "InterCloud" vision [5]. This approach could, however, reduce the protection of confidential data, as data may have to be replicated to multiple providers, and an agent may need to have the rights to move it without the owner's intervention. In such a case, a Quality of Protection should be part of the SLA.

The work presented here is the joint experience of two EU-funded projects. Contrail [4] is reusing an SLA framework from a third (earlier) project called SLA@SOI [3]; the OPTIMIS SLA [2] is based on the WSAG4J framework. Contrail is building IaaS federations which can manage cloud resources from multiple providers. OPTIMIS is managing IaaS provider attributes, to orchestrate applications (which may be legacy applications) on the cloud, specifically aiming to manage trust, risk, eco-friendliness, and cost as service parameters.

2 Existing Work

Grid information services are based on LDAP: originally Globus MDS [21], they have evolved into schemata known as GLUE [19] which can be expressed not just in LDAP but also in other formats (e.g., JSON or XML). Grid information services publish the state and availability of the service at some (hopefully recent) point in time, and dynamic data is refreshed regularly; they can be used as non-guaranteed service discovery as well as accounting.

Ruiz-Alvarez and Humphrey [1] developed a framework for describing cloud storage providers: the published information allows a customer to select services based on attributes. Like the grid, it is an instantaneous snapshot of the attributes, valid at the time it is created, and, as cloud service providers do not generally publish this information, requires some other party to update the information from time to time. Formatted in XML, the document includes a description of the providers' interfaces, security, geographic location, as well as samples of past measurements of performance metrics.

Indeed, once we move into negotiated cloud services, a number of requirements and failure modes can be recognised [20], Sec. 3.9, some of which are familiar from distributed computing (transient errors, credential timeouts), some of which could arise in heterogeneous distributed environments (inconsistent naming of capabilities, interoperation problems), and some may be specific to the cloud (the offer will need to be timely and accurate.)

Service discovery and matchmaking are well studied concepts in distributed systems, and a full survey of this work is beyond the scope and the focus of this paper. However, some of the experiences in matchmaking are equally valid for the present work. For example, [9] suggested the need for well-defined semantics, a common possibly extensible framework for agreement/negotiation, the need

to accommodate heterogeneity in services, the need for sufficiently expressive queries/matching, and perhaps allowance for a certain "fuzziness" in matching. Like the much earlier [12], the use of semantic web was proposed.

In the Italian Cloud@Home research project [14], the resource is opened not to a specific project but to a possibly unknown customer. More recently, the project looked at the use of SLAs [15]: the QoS for negotiation is based on monitored availability and estimated performance. Potentially, the provider could be an "at home" user (as with BOINC [16]), but the project currently only looks at spare resources in private clouds.

The "InterCloud" project [5] further investigated SLAs for QoS, but mostly aiming for execution time in PaaS [8]. This work relied on an estimation of the execution time of individual "jobs" along with an overall view of the expected number of jobs.

Service Oriented Architecture (SOA) models may look at QoS for web services (e.g., WS-ReliableMessaging) [17], Chap. 13. If we are orchestrating complex services from multiple cloud providers, the lessons of providing QoS with reliable web services becomes relevant. This orchestration of services is, however, at a higher level compared to the approach we take in this paper, and could usefully be the subject of future investigations. More generally, there has been work on federation of clouds, some of which will be touched upon in this paper. NIST is very active in relevant cloud activities [18] and OGF, DMTF, and SNIA are working on standards to enable federations.

3 Use of Standards

WS-Agreement [13] and WS-Agreement Negotiation [11] are open standards from the OGF [10]. To define or contribute to a new standard, projects need to start early, so the work on the standard has a chance of completing during the lifetime of the project. The use of open standards bodies is encouraged so other parties have a chance to contribute, or help test interoperability. While using a standard is no guarantee for interoperability, it reduces the complexity and effort to get two code-independent implementations of, e.g. a SLA negotiation, to successfully cooperate.

4 Federation and SLAs

4.1 Federation Model in Contrail

A Federation in Contrail is a software infrastructure managing a set of independent IaaS providers (but could also encompass storage as a service). Providers are typically competing private organisations, with their services differentiated along the three major axes of price, Quality of Service (QoS) and Quality of Protection (QoP).

Contrail services can be applied also when providers collaborate, such as community clouds, or when several datacentres are connected to the same federation to form a big private cloud. The Contrail federation can thus also be an enabler for Enterprise-level private clouds.

The software infrastructure offered by Contrail is structured into three main levels: *federation*, *provider* and *resource*. Briefly, the resource is the actual IaaS (or other resource), the provider is an abstraction with a common interface which publishes services and serves as an endpoint for SLA negotiations, and the federation provides the overarching control, aiming to realise the "Multi-Cloud" or "Cloud of Clouds" visions [5], [6]. In SLA Management at federation level, user requirements are expressed in a uniform way, independently of the underlying resources, thus enabling scenarios such as Cloud Brokering, Cloud Aggregation and cross-Cloud SLA Enforcement.

Once requirements are defined, the federation looks at available providers and negotiates with them. Actual resources (such as IaaS) are then provided at the resource level (so the provider must know how to authenticate to the resource on behalf of the user), and whenever possible, the provider level monitors the service provided by the resource.

4.2 OVF and SLA for Describing Complex Infrastructures

In Contrail, users can define an "*application*," an infrastructure configuration made up of one or more layers of Virtual Machines (VMs) connected by virtual networks. Applications use OVF, an open standard, to define the resources required for the application. An SLA then refers to a specific OVF descriptor file and expresses guarantees about specific OVF items, such as Virtual Systems. Each OVF VirtualSystem in Contrail specifies a VirtualMachine type (or template) of which multiple instances can be created at provisioning time. The same SLA document can express different guarantees for different items described in the OVF. For example the same SLA can specify that instances of VirtualSystem A must be located in UK and instances of VirtualSystem B must be located in France, thus making it possible to build a disaster recovery system based on two distinct geographical sites.

4.3 Automatic SLA Negotiation and Dynamic Pricing

When the user has described their application using OVF and identified in the SLA proposal all the guarantees which are needed on each element, they can initiate a negotiation via the Contrail Federation. The system will propose a price for each part of the application and will let the user experiment by negotiating different configurations until the right balance between price and QoS is found (Fig. 1).

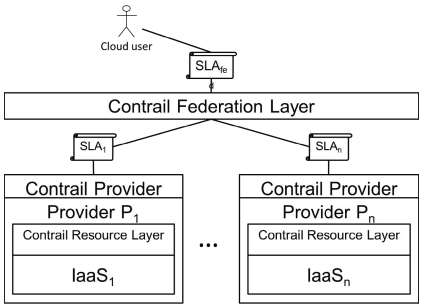

Fig. 1. Multi-level SLA negotiation

The provider layer automatically generates an SLA offer complete with a price for the resources described by the OVF and the QoS in the user's SLA proposal. The price for each VirtualSystem defined in the OVF depends not only on the amount of resources (e.g., memory or number of cores), but also on all the guarantees applied to it and identified in the SLA offer (e.g., location or co-location). The price is automatically calculated by the system for the required SLA+OVF configuration on the basis of the standard price list of that provider, but it could also be customised for the specific user by applying personalised marketing techniques. This personalisation helps the provider attract more customers, and weighing price/QoS helps users find the best value for money.

When a single provider cannot satisfy all the requirements, the federation can split the application and deploy it as an ensemble of cooperating parts on different providers. Splitting the application will go along with splitting the user's SLA proposal: the federation will negotiate each SLA part with different providers, and at each negotiation round will transparently provide to the user a SLA offer aggregating the best SLA offer for each of the parts. More on SLA splitting can be found in [7]. Further scenarios involving application and SLA splitting include scaling across multiple providers, Cloud bursting, and cross-provider SLA violation enforcement.

4.4 Proposed SLA Terms

Table 1 describes the QoS and QoP terms used (or planned) in Contrail SLAs.

QoS terms for IaaS providers can either express guarantees on the whole infrastructure, such as *availability*, or guarantees on a single resource, such as the *cpu_speed* of a specific VM. Most of the terms supported by Contrail are on single resources. When comparing SLA offers from different providers to find the best one, the value of some terms can be maximised (e.g. *storage_reliability*) or minimised (e.g. *price*), while the value of other terms must match exactly the user request in the SLA proposal. One such term expresses the geographical *location* of computing (VMs) or storage resources: in this case a SLA offer to be selected must match exactly the required countries

The term *Not_co_locate_host*, applied to a single VirtualSystem, specifies that all the VMs created from that VirtualSystem must be scheduled on different hosts, thus paving the way for high availability configurations.

The new *minimum_LoA* is a QoP term that specifies the minimum Level of Assurance (in authentication) that the resource requires, or the user defines for others to access their resource. Sensitive data will require a higher LoA: the federation authentication code sets the user's LoA when they authenticate; the required minimum is set at negotiation time and is also transmitted as an attribute when the authorisation checks run. Access to resources is blocked if *LoA* $<$ *minimum_LoA*. LoA is discussed further in section 5.1.

Table 1. SLA terms in Contrail

SLA term	Description
vm_cores	number of cores assigned to a VM
memory	RAM size assigned to a VM
cpu_speed	CPU frequency assigned to a VM
cpu_load	average system load of a VM over a 5-minute period
location	country code where the VM is located
availability	% of uptime of the whole provider infrastructure
location	country code where the given shared storage volume is located
storage_reliability	indicates the number of replicas for the given shared storage volume
reserve	number of VMs to be reserved for the given VirtualSystem
co_locate_rack	all the VMs from the given VirtualSystem must be allocated on the same rack
not_co_locate_host	all the VMs from the given VirtualSystem must be allocated on different hosts
minimum_LoA	minimum Level of Assurance that a user wants to be required for accessing her own resources

4.5 Federation Model in Optimis

The OPTIMIS toolkit was developed to support two kinds of providers: the Service Provider (SP) and the Infrastructure Provider (IP). Consequently, the OPTIMIS toolkit supports two different "flavours" of federation:

- a multi-cloud, where the SP uses resources from different cloud providers to deploy a service. This federation is visible to the SP's customer when it selects a service of the SP.
- a cloud federation, which is organised by the IP as part of its internal business processes. Thus, the federation is transparent for the SP and for reasons of data protection SLAs need an explicit term to allow or disallow using IP-federated resources for a concrete service of an SP.

4.6 Role of SLAs in OPTIMIS

SLAs in OPTIMIS are used to describe the SP's requirements on the services offered by an IP [1]. Therefore the SLAs are more technical than those an end-user would see. The SP's requirements are gathered in an XML document, the *Service Manifest*, which is used by the IP to assess whether the requirements can be

[1] The OPTIMIS toolkit is not targeting end-users, thus, and end-user SLAs are not in the scope of OPTIMIS.

fulfilled, have to be negotiated, or cannot be fulfilled at that time. When requirements can be met, the IP converts the Service Manifest into an agreement template for the SP which is then used to create an agreement (possibly after a round of negotiations). The Service Manifest is composed of three parts, the manifest core, which is used for creating the SLA, and the IP and SP extensions. Figure 2 presents the overall structure of the manifest. For a detailed description of the Service Manifest and its API see [22].

If the SP is federating resources, it can negotiate directly with IPs to figure out which are the most suitable Alternatively, the SP can ask a *broker* to organise the federation on its behalf. The broker receives the manifest, selects the suitable IPs, and creates SLAs with all the selected IPs and the SP.

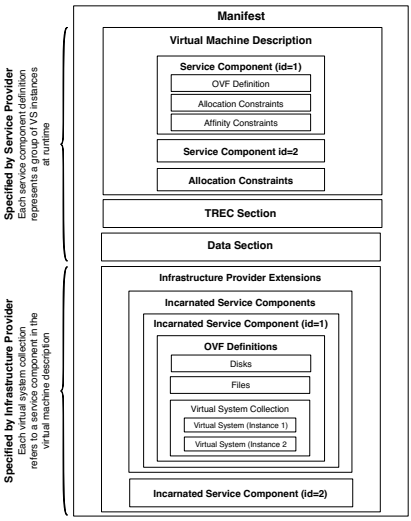

Fig. 2. Example of an infrastructure provider extension in the manifest

4.7 Proposed SLA Terms

The OPTIMIS SLA is based on WSAG4J which in turn is a a reference implementation of WS-Agreement and WS-AgreementNegotiation. Like Contrail, OPTIMIS both reused terms from other projects and developed its own. The latter are describe *trust*, *risk* (of SLA violation), *eco-efficiency* and *cost* (called TREC parameters). Trust is used like Contrail's LoA: An IP starting a federation selects other IPs with a trust level meeting the required trust level specified by the SP. A detailed discussion of the TREC parameters can be found in [23].

The pseudo-XML in Figure 3 shows examples of terms used in Service Manifests, the first showing elasticity, and the second showing eco-efficiency (as usual, '?' denotes that the preceding term is optional, '+' denotes one or more repetitions, and '*' zero or more.)

OPTIMIS also defined QoP terms, specifically data protection and data security, such as geographical locations of the data centres and required level and method for data encryption. Like Contrail, terms for co-location (or non-co-location) are used, which can require same/different rack or same/different datacentre.

5 Comparison and Discussion

The use of automated agents on the service provider side is a classic feature of the cloud (e.g., [18]): the provider does not need to negotiate with each customer

individually, so can provide services to a large number of customers, all of whom are treated the same. In turn, the provider saves money by providing the same services to many customers, by the economy of scale, at the cost of having to be elastic: the number of customers is not known beforehand.

The introduction of an automated agent acting on behalf of the users changes the cloud model. Ultimately, with an agent which is capable of discovering, negotiating, selecting, making use of, and combining services makes the consumer/provider relationship more dynamic. There are underlying assumptions: that the providers are interoperable and interchangeable, that they publish their SLA terms consistently and accurately, that they have consistent frameworks for authentication, authorisation, accounting, and monitoring. It may require relocatable services, or even a full federation built on top of the SLA negotiations, to enable users to access multiple providers with a single credential. But the potential gains are important.

```
<opt:ElasticitySection>
  <opt:Rule>
    <opt:Scope>
      opt:ScopeArrayType
    </opt:Scope>
    <opt:KPIName>xs:string</opt:KPIName>
    <opt:Window>xs:duration</opt:Window>
    <opt:Frequency>
      xs:positiveInteger
    </opt:Frequency>
    <opt:Quota>xs:positiveInteger</opt:Quota>
    <opt:Tolerance>
      opt:PositiveDecimal
    </opt:Tolerance>  ?
  </opt:Rule>  +
</opt:ElasticitySection>
<opt:EcoEfficiencySection>
  <opt:Scope>opt:ScopeType</opt:Scope>
  <opt:LEEDCertification>
    opt:LEEDCertificationConstraint
  </opt:LEEDCertification> ?
  <opt:BREEAMCertification>
    opt:BREEAMCertificationConstraint
  </opt:BREEAMCertification> ?
  <opt:EuCoCCompliant>
    xsd:boolean
  </opt:EuCoCCompliant> ?
  <opt:EnergyStarRating>
    opt:EnergyStarRating
  </opt:EnergyStarRating> ?
</opt:EcoEfficiencySection>
```

Fig. 3. Structure of terms used in Service Manifests

Smaller clouds – unlike the traditional cloud scenario, the customer/provider relationship can become more like the grid, where a single customer can use a significant proportion of the provider's capacity (of course, they don't *have* to, it's just that it is possible). The SLA fixes the customer/provider (or IP/SP) relationship and partially removes the need for elasticity, so the usage/capacity ratio does not need to be low. If a provider has an internal capacity of 100PB (say), and the consumer asks for 50PB, it doesn't matter that the consumer cannot double their capacity. In a federation, they can get the remaining capacity from a different provider. The provider knows that the consumer has allocated 50PB and can plan accordingly (for the duration of the agreement).

Potentially cheaper services – the provider can be less elastic than in a traditional cloud, and the planning ahead also means that the pricing of the service can be set lower, at least for long term agreements. Conversely, the customer's agent can look for lower prices from the competitors, and can even monitor the offers for spot pricing.

Better service delivery – as the agent can choose services from multiple providers, it can potentially combine these to obtain better QoS, while carefully tracking the QoP.

The potential gains are interesting, but should also be "reality checked." True mobility can be achieved only if services can be moved from one provider to another. A high level of interoperation is required. The requirements used by the user agents needs to be sufficiently expressive, but must not make it so complex that they won't be able to describe their requirements. Furthermore, the interplay between requirements needs to be better understood. And, above all, the publication of service terms needs to be timely, consistent, and accurate.

5.1 Quality of Protection

We have discussed a Quality of Protection, to be negotiated and agreed alongside the QoS. Unlike bandwidth, terabytes, or CPU hours, the terms in QoP are less well defined, although there are standards which can be built on. The terms of interest to our projects include:

LoA – the Level of Assurance associated with the user's credential. Covering this in depth is beyond the scope of this paper, but it needs to include factors like cryptographic strength, level of trust in identity provider, unique naming, multifactor authentication, usability, *etc.* Although multidimensional, the LoA is usually summarised in a single value. Typically, the *owner* of the resource or service will set the required minimum LoA based on an analysis of risks. Note that the negotiations of the SLAs themselves need a certain LoA, if we are to assert the trustworthiness of the agreement and the non-repudiation of it. With mutual authentication of sufficient LoA, the protection is also mutual: the customer is protected against a rogue provider, and the provider is protected against misuse by the customer because the customer has signed the agreement.

AUP – the Acceptable Use Policy could usefully be tied in with the SLA: by entering into the agreement, the customer asserts that they will comply with the rules and policies set out by the service provider(s). However, the challenge here is to define when it would be acceptable for an automated agent to enter into such an agreement: as before, the user-defined requirements will need to be clear enough that the agent can determine whether the proposed work is in violation of an offered AUP or not, and the offered AUP needs to be machine readable.

Data protection – again, a full discussion of data protection is beyond the scope of this paper, and, perhaps, beyond the capabilities of most non-specialised clouds. For now, the focus is primarily on availability (file is available at any given time) and integrity (it is not lost and not corrupted). Confidentiality is usually covered by encrypting in flight. Protecting personal data adds another level of difficulty: the best we can do in Contrail and OPTIMIS is to use geographic constraints, or have the providers advertise "no personal data."

6 Conclusion

In this paper we described two approaches for SLA-controlled Cloud federation in the two European projects FP7 projects Contrail and OPTIMIS. While the projects evolved independently using different frameworks for SLA creation and negotiation it turned out that the requirements regarding the SLA processing are similar and the underlying standards used are the same. Further work in the context of SLA negotiation will focus on achieving and testing interoperability of the two approaches.

Acknowledgements. This research was carried out as part of the Contrail and OPTIMIS research projects, funded by the European Commission under the 7th Framework Programme, grant agreements no. 257438 (Contrail) and 257115 (OPTIMIS).

References

1. Ruiz-Alvarez, A., Humphrey, M.: An Automated Approach to Cloud Storage Service Selection. In: ScienceCloud 2011, San Jose, California, USA, June 8 (2011)
2. Ferrer, A.J., et al.: OPTIMIS: A holistic approach to cloud service provisioning. Future Generation Computer Systems 28(1), 66–77 (2012)
3. SLA@SOI project homepage. Website, `http://sla-at-soi.eu/` (visited June 2013)
4. Contrail project homepage. Website, `http://contrail-project.eu/` (visited June 2013)
5. Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S.: Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability. In: Proc. of ICIW 2009, 4th Int'l Conf. Web App. and Services, pp. 328–336 (2009), doi:10.1109/ICIW.2009.55
6. Verissimo, P., Bessani, A., Pasin, M.: The TClouds Architecture: Open and Resilient Cloud-of-Clouds Computing. In: Proc. 2nd Int'l Workshop on Dependability of Clouds (DCDV 2012), together with IEEE/IFIP DSN 2012 (2012)
7. Contrail project, Deliverable D2.2, Architecture Design and QoS constraints matching algorithms in Federations (September 2011), `http://contrail-project.eu/documents/18553/136157/D2.2.pdf` (visited June 2013)
8. Buyya, R., Garg, S.K., Calheiros, R.N.: SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions, ArXiv 1201.4522 (2012)
9. Pantazoglou, M., Tsalgatidou, A.: A Generic Query Model for the Unified Discovery of Heterogeneous Services. IEEE Transactions on Services Computing 6(2) (2013)
10. Open Grid Forum. Website, `http://www.ogf.org` (visited May 2013)
11. Wäldrich, O., Battre, D., Brazier, F., Clark, K., Oey, M., Papaspyrou, A., Wieder, P., Ziegler, W.: WS-Agreement Negotiation Version 1.0. Technical Report GFD.193 (2011), `http://www.ogf.org/documents/GFD.193.pdf` (visited June 2013)

12. Svirskas, A., Wilson, M., Matthews, B., Arenas, A., Mac Randal, D., Gallop, J., Bicarregui, J., Lambert, S.: Towards an Efficient, Reliable and Collaborative Web: from Distributed Computing to Semantic Description, Composition and Matchmaking of Services, http://epubs.cclrc.ac.uk/bitstream/755/W3C-FSWS-CCLRCPositionPaper-Svirskas_mdw-v3[1].pdf (visited June 2013)
13. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Technical Report GFD.192 (October 2011), http://www.ogf.org/documents/GFD.192.pdf (visited June 2013)
14. Distefano, S., Puliafito, A.: Cloud@Home: Toward a Volunteer Cloud
15. Cuomo, A., Di Modica, G., Distefano, S., Puliafito, A., Rak, M., Tomarchio, O., Venticinque, S., Villano, U.: An SLA-based Broker for Cloud Infrastructures. J. Grid Comp. 11(1) (2013)
16. Graham-Rowe, D.: BOINC Puts Idle PCs to Work. New Scientist 180(2426-8) (2003) ISSN 0262-4079
17. Hewitt, E.: Java SOA Cookbook. O'Reilly (2009)
18. NIST Standards Acceleration to Jumpstart Adoption of Cloud Computing, http://www.nist.gov/itl/cloud/sajacc.cfm (visited June 2013)
19. Andreozzi, S., Burke, S., Field, L., Galang, G., Konya, B., Litmaath, M., Millar, P., Navarro, J.P.: GLUE Specification 2.0, Open Grid Forum GFD.147, http://www.ogf.org/documents/GFD.147.pdf (visited June 2013)
20. Badger, L., Bohn, R., Chandramouli, R., Grance, T., Karygiannis, T., Patt-Corner, R., Voas, J.: Cloud Computing Use Cases. NIST ITL Cloud Computing, http://www.nist.gov/itl/cloud/use-cases.cfm (visited June 2013)
21. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: Proc. 10th IEEE Int'l Symp. on High-Performance Distributed Computing, HPDC 2010 (August 2001)
22. OPTIMIS Service Manifest and API, http://www.optimis-project.eu/content/service-manifest-scientific-report (visited May 31, 2013)
23. Lawrence, A., Djemame, K., Wäldrich, O., Ziegler, W., Zsigri, C.: Using service level agreements for optimising cloud infrastructure services. In: Cezon, M., Wolfsthal, Y. (eds.) ServiceWave 2010 Workshops. LNCS, vol. 6569, pp. 38–49. Springer, Heidelberg (2011)
24. Kyriazis, D.: Cloud Computing Service Level Agreements. Technical report, European Commission DG CONNECT, Unit E2 - Software and Services, Cloud (July 2013), http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=2496 (visited July 15, 2013)
25. cybula - High performance pattern recognition systems. Website, http://www.cybula.com (visited June 30, 2013)
26. CompatibleOne-The Open Source Cloud Broker. Website http://www.compatibleone.org (visited June 30, 2013)