Using SLA Mapping to Increase Market Liquidity

Marcel Risch[1], Ivona Brandic[2], and Jörn Altmann[1]

[1] TEMEP, Department of Industrial Engineering, College of Engineering
Seoul National University, 599 Gwanak-Ro, Gwanak-Gu, 151-742 Seoul, South-Korea
marcel.risch@temep.snu.ac.kr, jorn.altmann@acm.org
[2] Information Systems Institute, Vienna University of Technology
Argentinierstr. 8/184-1, 1040 Vienna, Austria
ivona@infosys.tuwien.ac.at

Abstract. Research into computing resource markets has mainly considered the allocative fairness of market mechanisms. It has not been discussed how a large variety of resource types influences the market liquidity. Markets containing large numbers of buyers and sellers for heterogeneous resources suffer from a low likelihood of matching offers and requests. Traders therefore have the high risk of not being able to trade resources. We suggest a solution that derives SLA templates from a large number of heterogeneous SLAs in the market and, by using these templates instead of the original SLAs, facilitates SLA mapping. The usefulness of this approach is demonstrated through simulation results and a comparison with an alternative approach, in which SLAs are predefined.

Keywords: Service level agreements, Cloud computing, commodity goods, SLA matching, market liquidity, market mechanisms, SLA matching.

1 Introduction

When developing markets for Cloud resources, it should be considered that the Cloud resources are designed to be liquid. Illiquid goods (which include rare goods and differentiated goods) have a higher risk that they cannot be sold or purchased when needed, driving market participants away [19]. Therefore, markets can only function efficiently with a sufficient number of market participants.

Creating such a market with a large number of Cloud resource traders is far from trivial. Firstly, consumers will only join an open Cloud market, if they are able to find what they need quickly. Secondly, providers will only join the market if they can be fairly certain that the resources they are trying to sell will be sold for the right price. Should either of these conditions not be met, providers and consumers will not participate in the market.

Open Cloud markets face an unusual challenge in this instance. The widespread use of virtualization enables resource providers to create a wide range of tradable resource types. At the same time, resource consumers can also specify their needs very precisely. In such a case, this lack of liquidity will make the market unattractive to consumers and providers.

In this paper, we demonstrate the problem (low liquidity for each resource type) caused by a large number of resource definitions. To counteract this problem, we

A. Dan, F. Gittler, and F. Toumani (Eds.): ICSOC/ServiceWave 2009, LNCS 6275, pp. 238–247, 2010.

introduce an approach, based on SLA mapping, which ensures sufficient liquidity in the market. These SLA mapping techniques not only simplify the search for similar offers but also allow us to derive public SLA templates from all existing offerings. These SLA mappings map parts of a consumer-defined SLA document to a public SLA template. The purpose of SLA mappings is twofold: Firstly, users may discover services with less effort and secondly, based on predefined learning functions and accumulated SLA mappings; the proposed SLA mapping approach facilitates user-driven definitions of public SLA templates, increasing the chances of matching of offers in the future.

Summarizing, the contributions of this paper are: (1) the demonstration of the problem caused by a large number of resource types; (2) an approach to overcome the shortcomings of an open market by introducing an SLA mapping approach; and (3) a first evaluation of the proposed SLA mapping approach considering different strengths and weaknesses.

2 Computing Resource Markets

The research into resource markets can be divided into two groups, when looking at their attempts of describing the tradable good. The first group does not define goods at all, while the second group focuses on one aspect of a computing resource only. However, neither group discusses the liquidity of goods in Cloud computing markets.

The first group consists to a large extent of early Grid market designs [1-3, 8]. One such example can be seen in [1], where Buyya et al. describe the entities of a Grid market. However, the question of how the market mechanism can handle a multi-dimensional good has not been answered. Similarly, GRACE [2] is a market architecture for Grid markets and outlined a market mechanism without defining the good "computing resource" or considering that consumers and providers have to agree to the resource specifications.

The second group of Grid market research has simplified the computing resource good. When developing the MACE exchange [8], the authors recognize the importance of developing a definition for the tradable good and abstract computing resources into services which can be traded. However, a detailed specification of a computing resource service has not been given and hence its effects on market liquidity cannot be assessed. The Spawn market was envisioned to work with CPU time slices [9]. While matching demand and supply is trivial, the fact that the resources have been reduced to a single component is not realistic, as the CPU slice requirements depend on the CPU vendor due to different instruction sets. Lastly, the Tycoon market was developed before virtualization tools (e.g. Xen [11]) became widely used [10]. Initially, it worked with basic computing cycles and it was planned to extend this market by making use of virtualization. However, it seems that the effort has been discontinued. The SORMA project focused on fairness and efficient resource allocation [3-6]. The project identified several requirements for Grid markets [7]. However, the requirement analysis has not considered that a market can only function efficiently, if there is sufficient liquidity of goods.

Overall, much of the computing resource market research has worked with either simplified definitions of the tradable good or without defining the good at all. This

means that the issue of maintaining a sufficiently high liquidity (i.e. the likelihood of matching bids and asks) in such a market has not been addressed.

Despite some lacking research, a large number of commercial Cloud providers have entered the utility computing market, offering various types of services. On the one hand, there are resource providers, such as Amazon (e.g. EC2 [12]) and Tsunamic Technologies [13], who provide computing resources. Next, there are providers, who not only sell their own resources but also their own software services, such as Google Apps and Salesforce.com [14-15]. Furthermore, there are companies that attempt to run a mixed approach, i.e. they allow users to create their own services but also offer services themselves., e.g. Sun N1 Grid [16] , Salesforce.com, and Microsoft Azure [17]. Most of these providers have in common that they only sell a single type of resources (or, in the case of Amazon, resource types which are based on a basic instance [18]). This limited number of different resource types enables a market creation, since all demand is channeled towards very few resource types.

3 The Liquidity Problem in Markets

Virtualization makes the use of Cloud computing environments more interesting, since virtualization allows for portability of machine images. For a Cloud market, however, the introduction of virtualization introduces the problem that providers can define their own resource types to differentiate themselves from other providers. The impact of this behavior on liquidity is shown with an example of a double auction market mechanism. This liquidity issue also exists for other market mechanisms.

3.1 Double Auction

There have been a number of proposals for using double auctions for Grid markets [21-23], since double auctions have shown their suitability for non-storable commodities, such as electricity [24]. Contracts (i.e. service level agreements) in the double auction market, that we sketch here, are traded as follows: First, traders send their bids and asks to the market. Second, using the continuous double auction matching procedure, the market attempts to match bids and asks. Once a match is made, both parties receive a contract (i.e. signed SLA) with each other's information. The advantages of using such a contract lie in the fact that simple bids and asks suffice for trading resources.

3.2 Matching Performance for a Double Auction

We have developed a simulation for a double auction market, for which we defined a range of tradable resource types. Any trader could decide at the start of the day, whether resources had to be purchased or could be sold. The demand generator created a normally distributed demand across all traders. The simulation was repeated with various numbers of traders and various numbers of resources types. Each simulation allowed traders to interact over a period of 500 simulated trading days. Using these parameters, we measured the percentage of matched bids to the total number of bids. This measure was chosen, since the absolute trading volume only has meaning, if it can be compared to historical data. Our results are shown in Figure 1.

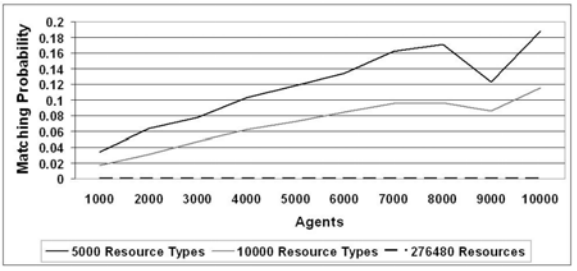

Fig. 1. Matching Probability for Bids

The top line shows the likelihood of matching a bid in a market with 5000 resource types. Initially, the matching probability is low, since the supply and the demand are spread out over many resource types and, hence, it is less likely that a buyer and seller will meet. However, the likelihood increases, as the number of traders in the market grows To determine how many traders would be needed to achieve a match probability of 75%, we performed a linear regression on the available data and found that about 39,800 traders would be needed. The grey line indicates the likelihood of matching a bid in a market with 10,000 resource types. Again, we can see that the probability is quite low but increases slowly with the number of traders. Using linear regression, we found that about 46,400 traders are needed to achieve a 75% match probability in this market. The dashed line indicates the likelihood of matching a bid in a market where each parameter of a resource can be set by traders, leading to 276,480 possible resource permutations. It can be seen that this probability is quite low and rises very slowly. A linear regression analysis indicated that about 33 million traders would be required to achieve a 75% matching probability for bids.

In general, these results show that, for a market to function properly, a sufficiently large base of traders must exist. Since large numbers of continuously acting traders are uncommon in current resource markets, our results indicate that a market, which sells fewer resource types, has higher liquidity. Next, we will show how resource types can be homogenized through SLA mapping, to get less resource types.

4 SLA Mapping

In this section, we explain the SLA mapping approach used to obtain public SLA templates, focusing on the lifecycle of SLA templates, management of SLA mappings, and SLA transformations used for the realization of SLA mappings. Finally, we introduce a marketplace, in which SLA mappings can be used to limit the number of resource types and, consequently, increase liquidity in the computing resource market. Note that automated trading is not required, purchases can be made manually.

4.1 The Importance of SLAs in Markets

Based on the outcome of the analysis of the liquidity problems in markets, we are faced with an interesting research challenge. On the one hand, to exploit the potential of open markets, a large number of traders is necessary. On the other hand, the large

number of traders inflates the variety of resources available. Without some form of SLA matching or some way of limiting resource diversity, the set of available computing resources in a market would grow, spreading supply and demand across a wide range of resources, thereby reducing the liquidity of each resource type.

Some current adaptive SLA matching mechanisms are based on OWL, DAML-S [25-26]. Another semantic technology work (by Oldham et al.) describe a framework for semantic matching of SLAs based on WSDL-S and OWL [16]. Dobson at al. present a unified quality of service (QoS) ontology, which is applicable to specific scenarios such as QoS-based Web services selection [24]. Ardagana et al. present an autonomic Grid architecture with mechanisms to dynamically reconfigure service center infrastructures to fulfill varying QoS requirements [27]. Koller et al. discuss autonomous QoS management, using a proxy-like approach for exploiting SLAs to define certain QoS parameters that a service has to maintain during its interaction with a specific customer [28].

None of the presented approaches address the issues of the open market, where traders meet on demand. In most existing approaches, traders have to agree either on specific ontologies [26-27] or have to belong to a specific portal [28]. None of the discussed approaches handle semi-automatic definitions of SLA mappings, allowing negotiations between inconsistent SLA templates. Also, none of the presented approaches allow user-driven definitions of publicly available SLA templates.

4.2 The SLA Template Lifecycle

The following figure illustrates the lifecycle of SLA templates. In particular, it shows how public SLA templates are generated and managed.

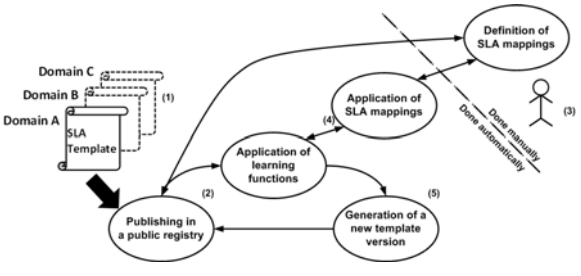

Fig. 2. SLA Template Lifecycle

Figure 2 shows how an initially generated SLA template (step 1), which may not necessarily reflect the needs of users from a specific domain, can be used to derive realistic templates for a new specific domain. As indicated through step 1, we assume that for specific domains, specific SLA templates are initially generated. These generated SLA templates are published in the public registry (step 2). Thereafter, SLA mappings are defined manually by users (step 3). At the same time, learning functions for the adaptation of these public SLA templates are defined. A learning function determines how often a specific SLA mapping had to be used during a predefined time period before the modified SLA template becomes a template stored in the repository (step 5). In such an environment, the application of SLA mappings can also be done automatically, as described in Section 4.3 (step 4).

4.3 Managing SLAs

Figure 3 shows the architecture for managing SLA mappings. The registry comprises different SLA templates where each of them represents a specific application domain, such as medicine or telecommunication.

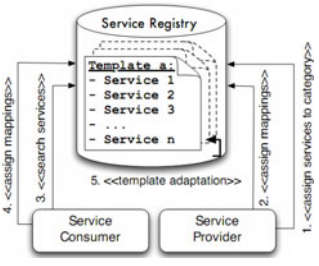

Fig. 3. Template Registry

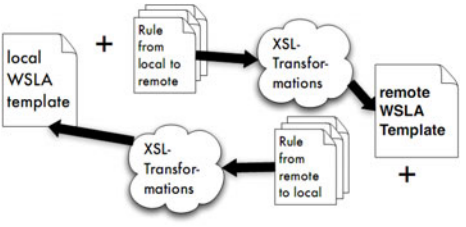

Fig. 4. SLA Transformation

Service providers may assign their service to a particular public SLA template (step 1 in Figure 3) and, if necessary, assign SLA mappings afterwards making modifications to the original template (step 2). Service consumers may search for the services using meta-data and search terms (step 3). After finding appropriate services, each service consumer may define mappings (i.e. modifications) to a posted template (step 4). Besides, public SLA templates should be updated frequently according to predefined adaptation rules to reflect the actual SLAs used by traders (step 5). To reduce the mapping cost for traders though, SLA mappings should also be executed automatically and public SLA templates should be defined and deleted dynamically.

4.4 SLA Mapping

To explain the SLA mapping, we describe how the transformation from modified public SLAs to the final SLA template is conducted. Figure 4 depicts a scenario for defining XSL transformations of SLAs expressed in the Web Service Level Agreements (WSLAs) specification language.

Public SLA templates are published in a searchable registry, from which traders may download and compare them with their local SLA template. If any differences are discovered, traders may write rules for transforming (XSL transformation) the local WSLA template to the remote template. The rules are stored in the database and can be applied at runtime to the remote WSLA template.

Figure 4 shows how negotiations can be performed using SLA transformation. Each of the two parties generates a WSLA. The locally generated WSLA plus the rules defining transformations from the local WSLA to the remote WSLA deliver a WSLA, which is compliant with the remote WSLA. Reversing this process, the remote WSLA plus the rules defining transformations from the remote to the local WSLA deliver a WSLA, which is compliant to the local WSLA. Thus, the negotiation may be done between non-matching WSLAs in both directions: from service consumer to service provider and vice versa. Thus, both parties may match on a publicly available WSLA template. To facilitate matching, SLAs are transferred into a canonical form as described in [29].

For example, in a double auction market, traders generate their own WSLA plus the rules for transforming their local WSLA to the market WSLA (i.e. the WSLA of the auction marketplace). The market has a learning function which can determine which of the WSLAs best matches the requirements of the traders. Based on this result, new public WSLAs can be created or old ones can be removed.

4.5 Application of the Public SLA Template in Double Auction Markets

In a double auction market environment, the market participants trade using public SLA templates, which describe the traded computing resources, the software running on these resources, the Terms of Use, and the price [30]. In Figure 5, a graphical representation of such a tradable public SLA template is shown.

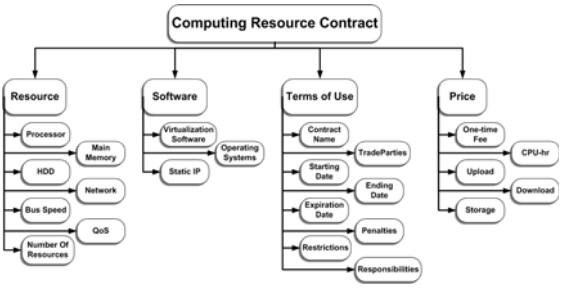

Fig. 5. A Sample SLA Template Obtained from a Learning Function

A market trading only a small number of public SLA templates encourages consumers to map their demand to the existing SLA templates, and providers to map their supply to the public SLA templates. However, once the market is operating, provisions can be made to add additional SLA templates (as new resources become available) and to remove SLA templates. Two rules are suggested to address this issue: First, a new tradable SLA is to be added, if the learning function determines that there is sufficient level of demand for this type of SLA template. Second, the removal of SLA templates depends on the trading volume: If the traded volume falls beneath a threshold for a certain time period, the SLA template will be removed.

Trading of SLA templates has to take into account that specific starting times and ending times are parameters of the templates. Standardization of starting times and usage durations provides one solution. Standardization for usage durations is fairly trivial, since it will become apparent that certain resources are commonly used for certain durations and less popular for other durations. Focusing on these popular periods will be sufficient for successful standardization. The learning function described previously can be used to determine popular starting and ending times.

5 Discussion and Validation

The following table gives an overview of the advantages and disadvantages of the SLA Mapping approach compared to an approach, in which the SLAs are predefined.

Table 1. Strengths and Weaknesses of SLA Mappings and Standardized SLAs

Approach	Strengths	Weaknesses
SLA Mapping	- No explicit resource limitations - Flexible SLA usage - Increased liquidity - User-driven approach	- SLA translations are time intensive - System abuse by submitting fake SLA mappings
Standardized SLA Templates	- Clearly defined resources - Adaptability of Standardized SLAs - Increased liquidity	- Standardized SLAs may prevent users with unusual resource profiles to join the market

The advantage of the SLA mapping approach is that users face few limitations if they determine their resource needs. They can define their SLA templates flexibly. The SLA transformation then determines those public SLA templates that minimize the differences to all user-defined SLAs. However, the SLA translations are time-intensive, which can be a problem if the SLAs are large. Furthermore, users can submit fake SLA mappings which have to be detected by a learning function.

5.1 Consequences of Few Resource Types

Figure 7 shows the gains that can be achieved by transforming many heterogeneous SLAs into a few public SLA templates, which can be traded in a double auction. This figure was obtained using the same simulation environment, which was used to demonstrate the liquidity problems earlier.

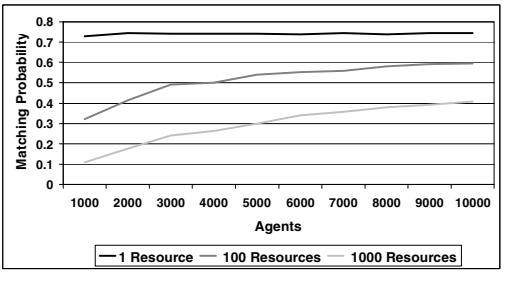

Fig. 7. Matching Probability for Bids with Fewer SLA Templates

The top line shows the likelihood of matching a bid for a market with one tradable SLA template. The likelihood is high (75%), since supply and demand is channeled to a single resource type. The next lower line shows the likelihood of a bid being matched for a market with 100 tradable SLA templates. Using linear regression, we calculated that a market with about 12,000 traders would have a matching probability of 75%. The lowest line shows the likelihood of a bid being matched in a market with 1000 tradable SLA templates. Using linear regression, we have calculated that, to reach a matching probability of 75%, 17,000 traders would be needed.

The matching probability of about 75% for a market with a single resource type seems quite low. The main cause of a lower matching probability was that bids had to

be satisfied by a single ask, since multi-tier applications usually must be closely co-located. The impact of duration and number of resources has been outlined in [20].

Concluding, double auction markets for computing resources require lots of standardization. Having many similar resource types distributes demand and supply, thereby threatening the liquidity. If resources are standardized, supply and demand can be channeled to fewer resources. Despite the reduction in overall utility caused by standardizing, standardization will increase liquidity. Our proposed SLA mapping approach creates standardized goods that can be traded in computing markets.

6 Conclusion

We have demonstrated that a market trading diverse computing resources can suffer from a reduced market liquidity. Since this problem makes a market extremely unattractive to traders, we have developed an approach to ensure that the market liquidity remains high. Our approach allows finding public SLA templates that are closest to the SLA templates defined by the user, therefore meeting the needs of users best. This approach is very flexible, since it adapts to the changing needs of the users. We also described the rules established within our approach to ensure that new SLA templates can be added and old SLA templates can be removed. We believe, based on the simulation results and the comparison with an alternative approach, that our approach will prove to be valuable in the development of an open Cloud market and will ensure sufficient liquidity to attract traders.

References

1. Buyya, R., Vazhkudai, S.: Compute Power Market: Towards a Market-Oriented Grid. In: First IEEE International Symposium on Cluster Computing and the Grid, CCGrid, p. 574 (2001)
2. Buyya, R., Abramson, D., Giddy, J.: An Economy Grid Architecture for Service-Oriented Grid Computing. In: 10th IEEE International Heterogeneous Computing Workshop, HCW 2001. IEEE Computer Society Press, Los Alamitos (2001)
3. The SORMA project (2009), http://sorma-project.org/
4. Nou, R., Julia, F., Guitart, J., Torres, J.: Dynamic Resource Provisioning for Self-adaptive Dynamic Heterogeneous workloads in SMP Hosting Platforms. In: International Conference on e-Business (2008)
5. Amar, L., Mualem, A., Stoesser, J.: On the Importance of Migration for Fairness in Online Grid Markets. In: International Conference on Autonomous Agents and Multiagent Systems (2008)
6. Amar, L., Barak, A., Levy, E., Okun, M.: An On-line Algorithm for Fair-Share Node Allocations in a Cluster. In: IEEE International Symposium on Cluster Computing and the Grid, CCGRID (2007)
7. Neumann, D., Stoesser, J., Weinhardt, C.: Bridging the Grid Adoption Gap – Developing a Roadmap for Trading Grids. In: Bled eConference, Merging and Emerging Technologies, Processes, and Institutions (2007)
8. Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: Trading Grid Services - A Multi-Attribute Combinatorial Approach. European Journal of Operational Research 187(3), 943–961 (2008)

9. Waldspurger, C.A., Hogg, T., Huberman, B.A., Kephart, J.O., Stornetta, W.S.: Spawn: A Distributed Computational Economy. IEEE Transactions on Software Engineering 18(2), 103–117 (1992)
10. Lai, K., Rasmusson, L., Adar, E., Zhang, L., Huberman, B.A.: Tycoon: An Implementation of a Distributed, Market-Based Resource Allocation System. Multiagent Grid Systems 1(3), 169–182 (2005)
11. XenSource, Inc. (2009), http://xen.org/
12. Amazon Elastic Compute Cloud (Amazon EC2) (2009), http://aws.amazon.com/ec2/
13. Tsunamic Technologies Inc. (2008), http://www.clusterondemand.com/
14. Google Apps (March 2009), http://www.google.com/apps/
15. Salesforce.com (March 2009), http://www.salesforce.com
16. Sun Grid (2009), http://www.sun.com/service/sungrid/index.jsp
17. Microsoft Azure (2009), http://www.microsoft.com/windowsazure/
18. Amazon EC2 Instance Types (2008), http://aws.amazon.com/ec2/instance-types/
19. Samuelson, P.A., Nordhaus, W.D.: Economics, 18th edn. McGraw-Hill/Irwin (July 2004), ISBN 0072872055
20. Altmann, J., Courcoubetis, C., Stamoulis, G.D., Dramitinos, M., Rayna, T., Risch, M., Bannink, C.: GridEcon - A Market Place for Computing Resources. In: Altmann, J., Neumann, D., Fahringer, T. (eds.) GECON 2008. LNCS, vol. 5206, pp. 185–196. Springer, Heidelberg (2008)
21. Weng, C., Lu, X., Xue, G., Deng, Q., Li, M.: A Double Auction Mechanism for Resource Allocation on Grid Computing Systems. In: Jin, H., Pan, Y., Xiao, N., Sun, J. (eds.) GCC 2004. LNCS, vol. 3251, p. 269. Springer, Heidelberg (2004)
22. Kant, U., Grosu, D.: Double Auction Protocols for Resource Allocation in Grids. In: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2005), vol. 1(1), pp. 366–371. IEEE Computer Society, Washington (April 2005)
23. Weng, C., Li, M., Lu, X.: Grid Resource Management Based on Economic Mechanisms. Journal of Supercomputing 42(2), 181–199 (2007)
24. European Energy Exchange (2008), http://www.eex.com/en/
25. Dobson, G., Sanchez-Macian, A.: Towards Unified QoS/SLA Ontologies. In: Proceedings of the IEEE Services Computing Workshops, SCW 2006, Chicago, Illinois, USA, pp. 18–22 (September 2006)
26. Oldham, N., Verma, K., Sheth, A.P., Hakimpour, F.: Semantic WS-Agreement Partner Selection. In: Proceedings of the 15th International Conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK (May 2006)
27. Ardagna, D., Giunta, G., Ingraa, N., Mirandola, R., Pernici, B.: QoS-Driven Web Services Selection in Autonomic Grid Environments. In: International Conference Grid Computing, High Performance and Distributed Applications, GADA 2006, France (November 2006)
28. Koller, B., Schubert, L.: Towards Autonomous SLA Management Using a Proxy-Like Approach. In: Multiagent Grid Systems, vol. 3(3). IOS Press, The Netherlands (2007)
29. Brandic, I., Music, D., Dustdar, S.: Service Mediation and Negotiation Bootstrapping as First Achievements Towards Self-Adaptable Grid and Cloud Services. In: Grids Meet Autonomic Computing Workshop, GMAC 2009. In conjunction with the 6th International Conference on Autonomic Computing and Communications, Barcelona, Spain (June 2009)
30. Risch, M., Altmann, J.: Enabling Open Cloud Markets Through WS-Agreement Extensions. In: Service Level Agreements in Grids Workshop, in conjunction with GRID 2009, Banff, Canada. CoreGRID Springer Series (October 2009)