CONTRACT SERVICES FOR POST-DISCOVERY GUARANTEE MANAGEMENT

Josef Spillner, Bastian Buder, Torsten Schiefer and Alexander Schill
TU Dresden, Nthnitzer Str. 46, 01062 Dresden, Germany

Keywords: SOA, SLA management, SLA management, Contracts.

Abstract: Contract services let service providers and consumers manage service level agreements (SLAs) and other contractual documents. These services let users negotiate SLAs, select tariffs, submit selected service offerings to auctions, view and manage running contracts and finally submit feedback about the satisfaction with the usage of contract-bound services. We define a number of contract services specifically for consumers which run as platform-level services with a common associated user interface called Contract Wizard. The increased interactivity over purely autonomous approaches is crucial for a wider acceptance of contract-bound service execution.

1 INTRODUCTION

Service discovery is the process of matching functional and non-functional consumer requirements against provider offers. Many interesting algorithms for the matchmaking exist, but most descriptions stop at the interesting question: Once a service is found, what can a consumer do with it? Technically, service discovery returns a service description and a communication endpoint, both of which can be used to invoke a service from another service or, if the search was initiated interactively by the consumer, integrate associated user interfaces into the client system or render a form dynamically. Sometimes, however, adhoc invocation is not desirable. Instead, guarantee terms shall be negotiated so that the service consumer can rely on the offered properties.

Contracts can be established between service providers and service users to obtain verifiable and legally valid records of specific agreements regarding service quality or other potentially conflicting goals on the technical, business and legal level. According to (Truong et al., 2008), contract management involves several phases, including specification, negotiation and monitoring. Other approaches add renegotiation and feedback as additional aspects. Usually, research on this management is restricted to autonomous agents. However, in order to let the concept of contracts gain acceptance in service-oriented computing, concepts need to be developed to let users (both providers and consumers of services) control each phase.

In our approach, the web application Contract Wizard handles all contract-related matters from a consumer's point of view, including the initial negotiation of non-functional properties (NFPs), selection of tariff options and handover to auctioning platforms in the negotiation phase. The monitoring and feedback phases are supported with a contract visualisation and management tool and the collection of feedback regarding the satisfaction with the contract, respectively. The contract services implementing this functionality are seen by us as a set of useful, standardisable parts of service brokering architectures. In this paper, we first describe the usefulness of contract services and report on existing approaches of contract management. We present a reasonable set of them on a conceptual and logically linked level. To illustrate the concept, we then show to which extent our current prototype already improves the user experience on service marketplaces. Finally, we report on open issues which need to be solved before contract services can eventually replace inflexible manual contract management approaches.

2 PROBLEM STATEMENT

In next-generation networks, service usage contracts – also known under the acronym SLA, for Service Level Agreements – are said to play an important role to ensure a high quality as well as technical, financial and juridical reliability (Braun et al., 2008). Such contracts usually contain clauses about minimum or maximum average values of technical characteristics, tolerance ranges, usage constraints and other restrictions. In addition, they may contain compensation rules. Finally, any contract will have a set of metadata such as the unambiguous identification of the participating partners – or parties, in legalese –, the duration of validity, associated tariff models and disclaimers. Without the metadata, the contract would just be an agreement without legal relevance. Often, the term SLA is used to refer to purely technical contracting approaches and Quality of Service (QoS) guarantees. On the other hand, formal service interface definitions as described by the Web Service Description Language (WSDL) or the older Interface Description Language (IDL) are also named contracts in the context of software engineering. Our understanding of contracts requires the mutual acceptance of an SLA by at least two involved parties.

Several declarative languages for expressing agreements and contracts have been created over the last decade. Among them, WS-Agreement, WSLA and SLA-ng have been subject to a considerable amount of analysis, critique and proposed improvements (Tian et al., 2004; Frankova et al., 2006; Truong et al., 2008). A number of service execution platforms have been specifically developed or extended to exploit the contents of such contracts and act accordingly in decisive situations where the contracts conflict with the state of the platform. Depending on the nature of the situation, either the contracts can be modified by renegotiating or cancelling them, the service execution can be modified by terminating, reconfiguring, rebinding or otherwise adapting services, and the platform can be modified by acquiring more resources, migrating workload to other servers or equivalent actions which help solve the conflict. These technical, corrective actions belong into the functionality of all adaptive platforms for contract-bound service execution. The term Service-Level Management (SLM) encompasses adaptivity but especially includes changes of a contract within its lifecycle as well as admission control functionality to reduce the need for adaptivity. An approach to SLM is proposed by (Wang et al., 2007) and refers to technical QoS management and admission control using a custom QoS specification language. Some of its goals like end-to-end QoS guarantees are feasible in closed environments but unsuitable for Internet-scale service delivery. Furthermore, no contract lifecycle guidance for users is offered by the system. In (Ludwig and Franczyk, 2006), current SLA languages are compared and evaluated for suitability for automated contract management in service grids. The authors conclude that several ambiguities are present in all of them which makes automated approaches hard or impossible for more complex conditions. When elastic infrastructures are used beneath a user-oriented participative Internet of Services (pIoS), user guidance through contract services becomes even more important.

While the autonomous handling of contracts is sufficiently covered by existing approaches, additional SLM aspects are needed for conveying the conflicts and decisions to the user in an intuitive and comprehensible way. No previous concepts are known at this point which could fill this gap. Hence, we introduce dedicated, reusable contract services as a novel concept. Furthermore, we propose Contract Wizard as an intuitive user interface to contract services in the context of pIoS.

3 CONTRACT SERVICES

Given that the processes of creating, storing, managing and terminating contracts encompass multiple steps, we propose a set of individual contract services which are loosely coupled so they can be used both as standalone platform services and as a composite service which manages all of the contractual aspects. The research is focused on initial, user-driven negotiation of SLA parameters (technical perspective), tariff selection and auctioning (business perspective), contract monitoring and status visualisation, interactive feedback collection and the adherence to contract law (legal perspective) in any of these steps. We do not consider contract services aimed at providers who place service offerings at marketplaces in this paper, although it can be assumed that the basic functionality would be comparable. Figure 1 shows the overall picture.

3.1 Initial SLA Negotiation

The scope of the initial negotiation is the user-driven transformation of a service description or an SLA template into an SLA. A set of templates is attached to a service description within the service registry, from which the user may choose one to initiate the negotiation process. If no templates are present, an unconstrained negotiation can be initiated. The resulting SLA will be linked to the user to form the base for a valid contract between two parties. The term *initial* serves to differentiate this step from autonomous renegotiations which may happen once the SLAs have become binding. It is assumed that only NFPs are

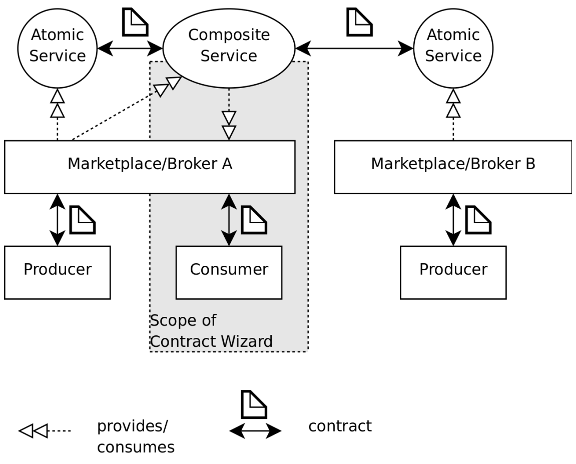


Figure 1: Contract relationships between pIoS hosters, service producers and consumers.

subject to negotiation, while functional properties are fixed from the moment of service selection. The extension to allow the contract peer to find functionally equivalent or similar services is not considered by us. The negotiation service runs as an agent acting on the user's behalf and representing the user's will. The negotiation peer would usually be provided by the service execution environment. In several projects, an SLA manager agent assumes this role. The transformation uses form generation techniques to create a user interface representing the possible options and NFPs for negotiation. It contains fixed values as well as variable properties which may be altered by the user. Alteration can happen either on a per-contract basis using roundtrips to the negotiation peer, or for more sophisticated needs and more intuitive understanding of inter-property dependencies on a per-property basis. Property ranges may be initially restricted and preset by importing the user's preferences from the service discovery process. At the moment, we are not aware of a standard file format for expressing and transmitting user goals. We propose to leverage the prioritised requirement/offer match from the discovery, which may be expressed as a semantic query in the Web Service Modelling Language (WSML). The preferences are expressed as either a quadruple consisting of property name, value, unit and comparison operator. Each preference item is thus transformed into a service level objective (SLO) in the contract as shown in figure 2.

3.2 Tariff Selection

Tariffs regulate how much a user has to pay to access a service. Every tariff is based on an economically viable pricing model which aims at maximising

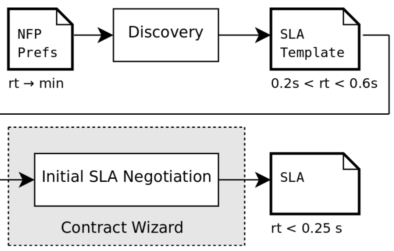


Figure 2: Preference-influenced negotiation example for response time (rt).

the income for the service provider while keeping the price competitive enough to not drive potential and current users to competitors. In addition, there is a mutual relationship between the applicability of tariffs and the set of outcomes of negotiation processes. For example, selecting a low-budget tariff restricts the guarantees on response time and encryption strength, whereas negotiating an agreement of near-100% uptime will lead to the expulsion of low-budget tariffs in subsequent tariff selections. Due to the loose coupling of contract services, either order of initial negotiation and tariff selection will be possible. Technically, either a reference to a tariff or the tariff terms themselves are added to the SLA.

3.3 Contract Monitoring and Management

While the creation of a contract only takes a limited amount of time in the range of minutes up to a few days, the contract validity period can often reach months or years. During this time, the user needs to stay informed about the set of running contracts (passive control). There should also be facilities for modifying, terminating or otherwise managing contracts (active control). It is generally expected that the user-centric Internet and user-driven actions like service composition or content creation will increase rapidly in the pIoS. Giving information and control back to the user is mandatory for the broad acceptance of these concepts.

The information gathered by the various monitoring sources in service execution environments can be used to present contract status information to the user. In particular, generated graphs related to negotiated properties as well as to the overall contract compliance of the service execution are of great help to decide whether a renegotiation or cancellation will be beneficial. Further information such as detailed lists of incidences can be retrieved through a Monitoring-as-a-Service (MaaS) interface to the marketplace monitoring database. Of course, the transparency and level of detail for such views can be restricted at the discretion of the hosting operator. There are various reasons for such restrictions, including the fear for providing hints on the used infrastructure to competitors and the risk of leaking data due to technical glitches. For the purpose of our research, we assume full access of users to data directly related to their contracts. Some of the features will hence not apply if restrictions are in place. Key relevant information displayed to the user ideally includes a timeline of values per monitored property, a timeline of incidents linked to explanations by the provider and statements about whether the user's specific contract has been affected, payment status information and overall average and current values. In practice, it again depends on provider policies of how much of this information will be available to the user.

3.4 Feedback Collection

While monitoring data gathered from service execution will be objectively accurate, there might always be situations in which the user's perceived quality of a service deviate from the measured value. A classic example is a fully-operational host with a broken connection to the user's computer. In such cases, additional feedback from the user regarding the fulfilment of the contract by the provider can be very helpful and might indeed protect the provider from greater damage of having to compensate a number of consumers when problems propagate. Such ratings provide innovative value over traditional, contract-agnostic approaches to automatic and user feedback on service execution (Maximilien and Singh, 2002).

3.5 Adherence to Law

In order to turn an agreement (or SLA) into a widely-recognised contract, the peculiarities of contract law need to be considered for the whole process of contract creation, modification and termination. Even in legally harmonised systems like EU member states, the acceptance and legal relevance of qualified digital signatures, electronic contracts and terms of service varies greatly. Therefore, we only consider the legal issues for a limited number of countries in our work on contract services. The issue is complicated further by the nature of consumers as either business users or private consumers. Already identified requirements encompass the prominent display of the court location of the provider company and double confirmation according to the Rome Convention of 1980 and its successor e-commerce laws, the unambiguous acting of Contract Wizard on behalf of and in full cooperation with the user, and the possibility to print contracts for postal delivery in case electronic contracts are not recognised by law (Parrilli, 2008). Users must be able to modify all parts of SLA offers before finalising them. Contracts must explicitly inform about potential renegotiation attempts in the future. Detailing these requirements and adding further legal safeguards is an ongoing process in the definition of useful and legally viable contract services. We plan to report on the outcome of these activities based on a cooperation and continuous review from researchers in the area of contract and civil law.

4 PROTOTYPICAL EVALUATION

We have implemented all mentioned contract services and consolidated them into a single application called Contract Wizard. The name reflects the user-centric step-by-step guidance nature of the application. In addition to the application itself, this section is going to present major integration points with additional brokering components, namely the service discovery for the initiation of contract creation, as well as the back-end services for all of the steps in Contract Wizard: The pricing models and corresponding pricing model repository, auctioning and brokering platforms, the usage feedback service, the monitoring history service, and finally the SLA manager. The architecture and workflow is shown in figure 3.

4.1 Contract Wizard

Contract Wizard is typically invoked in the matchmaking or brokering phase of any service usage lifecycle, directly after the service discovery step and before the service usage phase. It also extends into the usage and feedback phases. It is located on user-focused portals. The use of this application is aimed at service consumers. The application itself is implemented with eRuby and Java/JSP. It acts as a shell containing page modules for all the possible steps during the creation and lifetime of a contract. Each page module interacts with a platform service, which is in turn connected to a repository or another web service.

Page modules can be skipped if no choice is available. In particular, service providers may opt to not offer configurable SLA levels or tariffs, or only include fixed SLA templates without negotiable parts. In parallel to the active page module, Contract Wizard permanently displays a list of running contracts for easy access. The figures 4 and 5 demonstrate typical negotiation controls as seen by a user and a resulting SLA section.

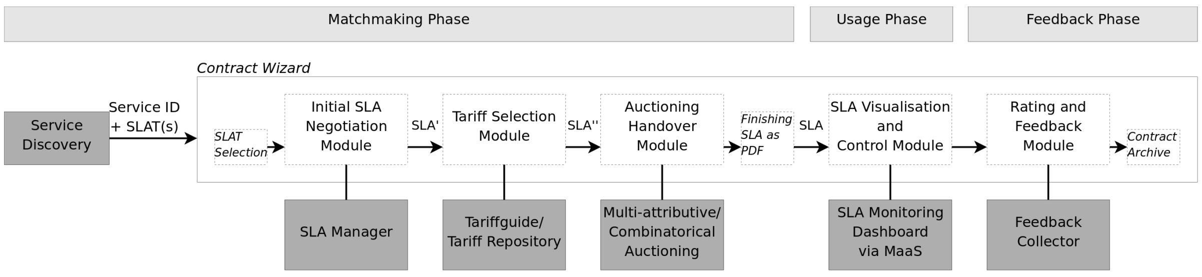
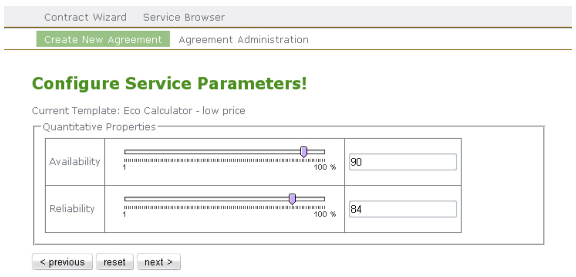

Matchmaking Phase
Usage Phase
Feedback Phase
Contract Wizard
Service Discovery
Service ID + SLAT(s)
SLAT Selection
Initial SLA Negotiation Module
SLA'
Tariff Selection Module
SLA''
Auctioning Handover Module
Finishing SLA as PDF
SLA
SLA Visualisation and Control Module
Rating and Feedback Module
Contract Archive
SLA Manager
Tariffguide/ Tariff Repository
Multi-attributive/ Combinatorical Auctioning
SLA Monitoring Dashboard via MaaS
Feedback Collector

Figure 3: Contract document workflow within Contract Wizard.



Configure Service Parameters!

Current Template: Eco Calculator - low price

Quantitative Properties

Availability	1 — 100 %	90
Reliability	1 — 100 %	84

Figure 4: Screenshot of initial negotiation in Contract Wizard.

```
wsag:GuaranteeTerm Name="Availability" {
 wsag:ServiceScope ServiceName="ExService";
 wsag:ServiceLevelObjective {
  wsag:KPITarget {
   wsag:KPIName "Availability";
   wsag:CustomServiceLevel "90";
}}}
wsag:GuaranteeTerm Name="Reliability" {
 wsag:ServiceScope ServiceName="ExService";
 wsag:ServiceLevelObjective {
  wsag:KPITarget {
   wsag:KPIName "Availability";
   wsag:CustomServiceLevel "84";
}}}
```

Figure 5: Resulting WS-Agreement SLO definition in compact notation.

Both the initial negotiation and the feedback modules are driven by the NFPs offered by the service. Our implementation supports configurable category systems so that most users can opt to only configure the most interesting properties, e.g., performance and price, while advanced users can negotiate all properties separately.

4.2 Service Discovery

Contract Wizard provides a direct initiation mode which requires the user to give it the URL to a service description and contract template. Usually though, service discoveries like ConQoMon link several actions to found services, including the option to invoke the service directly and to negotiate a contract with it (Stoyanova et al., 2008). Either way, the description and associated contract templates are fetched by Contract Wizard through HTTP GET or passed to it through HTTP POST and used to create the contract creation session.

4.3 Pricing Models Repository

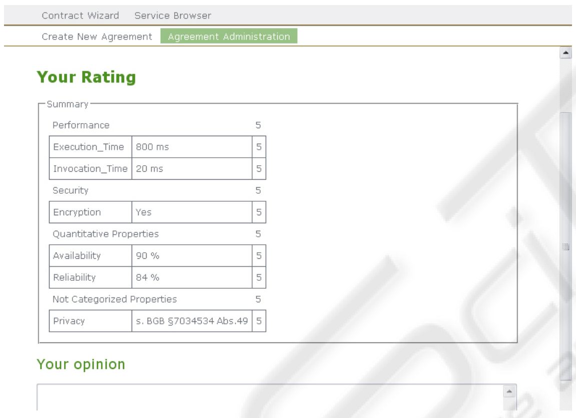
In order to increase the acceptance of autonomous service hosting, rather complete pricing models and taxonomies have been developed on an abstract level, such as (Lehmann and Buxmann, 2008). This model can be expressed in either XML or in ontology languages. Eventually, Contract Wizard's tariff selection will be based on this model. Currently, an older, less complex model derived from an implementation called WS-Accounting (Br, 2008) is used for this purpose. The tariffs are expressed in XML and stored in a file system repository. We have implemented a helper application called *tariffguide* which scans this repository and is able to select the best tariff based on criteria like the number of expected invocations. In addition, a web page lists the tariffs and offers them for selection and inspection of details. Both parts of the implementation are considered to be reusable once the more complex model will be used.

4.4 Auctioning Platforms

As soon as service users have several contracts with a provider, the optimisation potential moves from the modification of all single contracts to the improvement of the set of contracts. Multiple services might be offered for less cost than the sum of all of them when contracted individually, for example. In these situations, Contract Wizard allows the user to submit a negotiated offer together with the preferred tariff to an auctioning platform. Two such platforms for multi-attributive and combinatorial auctions (van Dinther et al., 2008) have been integrated on a conceptual level. The integration efforts are currently still ongoing and are based on HTTP references to contract offers and tariffs. During the auction, the Contract Manager does not proceed. Once an auction is finished, the respective auctioning platform invokes a re-entry page of Contract Wizard, therefore signalling it the results of the auction and triggering it to proceed to the contract finalisation step. Due to the expected delay until a contract becomes final and valid, the auction will only be applied to contract creation processes without realtime requirements. The combinatorial auction needs information about a set of services. This is realised by having the discovery expose the BPEL structure of composite service.

4.5 Service Usage Feedback

Explicit feedback and rating on the usage of contract-bound services is collected from the user through an automatically generated form. The contents of the form are submitted to the usage feedback aggregator. The form is shown in figure 6, with figure 7 explaining the process.



Your Rating

Summary

Performance		5
Execution_Time	800 ms	5
Invocation_Time	20 ms	5
Security		5
Encryption	Yes	5
Quantitative Properties		5
Availability	90 %	5
Reliability	84 %	5
Not Categorized Properties		5
Privacy	s. BGB §7034534 Abs.49	5

Your opinion

Figure 6: Screenshot of completed rating form in Contract Wizard.

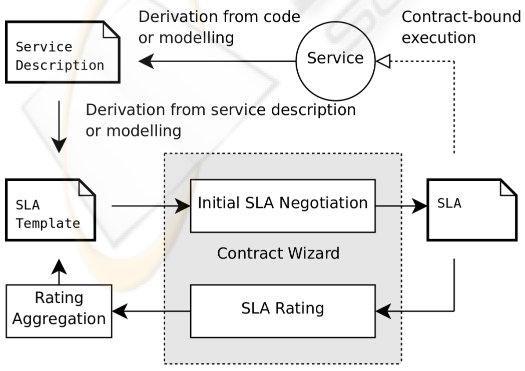


Figure 7: SLA negotiation and feedback loop.

4.6 Monitoring History

For the purpose of integrating the monitoring results into Contract Wizard, several possibilities exist. In order to display lists and tables, our prototype can connect to the Monitoring-as-a-Service interface and query for relevant data such as daily average response times. Displaying graphics, on the other hand, integrates the results of visualisation components which themselves use the MaaS interface. In most cases, Contract Wizard embeds references to already generated graphs.

4.7 SLA Management

While the SLA negotiation part of Contract Wizard turns an SLA template into an SLA offer, the task of the SLA manager component is to evaluate the offer and accept or reject it. The two services communicate with each other using a negotiation protocol which is the on-wire representation of a negotiation algorithm. There are a number of algorithms around, but only a subset can be used for this communication: Some algorithms only exist on paper without or with only insufficiently performing implementations. Likewise, some algorithms are bound to the SLA format in use. Avoiding these two subsets, the remainder is the junction of all SLA format-agnostic algorithms and WS-Agreement-based algorithms. The default negotiation mandated by WS-Agreement is known to be insufficient, though. In our work we closely watch the development of a successor with more modular split between file format and negotiation protocol, while at the same time looking into creating a generic negotiation toolbox with pluggable algorithms. That way, we can cover multi-phase negotiation, incremental constraints and other advanced algorithms.

5 OUTLOOK AND NEXT STEPS

In the future, we want to discuss the general idea of contract services and the specific implementation Contract Wizard within the wider research community. There is generally a high interest to treat SLAs as contractually valid contracts to enable the semi-automatic creation and management of contracts in real-world applications. Contract Wizard is already offered publically as part of our SOA platform *TEC*ЛA[1], in which it cooperates with other components, especially the service discovery. The

[1]TECЛA and Contract Wizard website: `http://texo.inf.tu-dresden.de`

focus of ТЕСЛА is to provide a well-integrated, open source platform which covers all important aspects regarding contractually guaranteed properties and quality. ТЕСЛА includes additional components, such as a dynamic invocation tool (Dynvoker), a context- and quality-aware semantic service discovery (ConQo) and another step-by-step web application aimed at service providers who want to publish services (Provider Wizard). We believe that both the theoretical foundations for a real-world adoption as well as a ready-to-use and freely available implementation are mandatory to raise awareness and acceptance for the idea of contract services.

6 CONCLUSIONS

Our research on establishing and managing contracts between service users and service providers has turned up a number of concepts for individual, yet loosely-coupled platform services which we call contract services. Aligned with the vision of a participative Internet of Services, we have identified the need to incorporate these concepts into the marketplace and link them to related research activities by various research groups concerned with contract law, pricing models, auctioning and brokering as well as SLA management. Therefore, we have introduced Contract Wizard, an essential user-visible service marketplace component for contract management. Its integration with semantic service discovery helps users to find suitable contracts based on initial preferences and custom negotiation.

ACKNOWLEDGEMENTS

The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference “01MQ07012”. The authors take the responsibility for the contents.

REFERENCES

Br, M. (2008). Steuerung der Nutzung von Web Services durch Vertrge und Bezahloptionen. Thesis, Technische Universitt Dresden.

Braun, I., Reichert, S., Spillner, J., Strunk, A., and Schill, A. (2008). Zusicherung nichtfunktionaler Eigenschaften und Dienstgte im Future Internet of Services. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 31(04/08):225–231.

Frankova, G., Malfatti, D., , and Aiello, M. (2006). Semantics and Extensions of WS-Agreement. *Journal of Software (JSW)*, 1(01):23–31.

Lehmann, S. and Buxmann, P. (2008). Preisstrategien fr Software- und Serviceanbieter. In *Schriften zur Quantitativen Betriebswirtschaftslehre*, number 4.

Ludwig, A. and Franczyk, B. (2006). SLA Lifecycle Management in Services Grid - Requirements and Current Efforts Analysis. In *Proceedings of the 4th International Conference on Grid Services Engineering and Management (GSEM)*, pages 219–246. Erfurt, Germany.

Maximilien, E. M. and Singh, M. P. (2002). Reputation and Endorsement for Web Services. *ACM SIGecom Exchanges*, 3(1):24–31.

Parrilli, D. (2008). SLAs and Legal Issues. In *Workshop on Infrastructure Services from a Business Perspective*. ServiceWave, Madrid, Spain.

Stoyanova, G., Buder, B., Strunk, A., and Braun, I. (2008). ConQo A Context- and QoS-Aware Service Discovery. In *Proceedings of IADIS Intl. Conference WWW/Internet*. Freiburg, Germany.

Tian, M., Gramm, A., Ritter, H., Schiller, J. H., and Winter, R. (2004). A Survey of current Approaches towards Specification and Management of Quality of Service for Web Services. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 27(03/04):132–139.

Truong, H.-L., Gangadharan, G., Treiber, M., Dustdar, S., and D'Andrea, V. (2008). On Reconciliation of Contractual Concerns of Web Services. In *2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC)*. Dublin, Ireland.

van Dinther, C., Holtmann, C., Setzer, T., Stage, A., and Stathel, S. (2008). Auctions for Service Brokerage in Business Value Networks. In *Group Decision and Negotiation*. Coimbra, Portugal.

Wang, C., Wang, G., Wang, H., Chen, A., and Santiago (2007). Quality of Service Contract Specification, Establishment, and Monitoring for Service Level Management. *Journal of Object Technology, Special Issue*, 6(11):25–44.