Dynamic SLA Template Adjustments based on Service Property Monitoring

Josef Spillner and Alexander Schill
Technische Universität Dresden
Faculty of Computer Science
Nöthnitzer Str. 46, 01062 Dresden, Germany
{josef.spillner,alexander.schill}@tu-dresden.de

Abstract

Service Level Agreements (SLAs) are used to manifest guarantees about certain functional and non-functional aspects of service execution. Service providers are confronted with a hard problem when trying to estimate reasonable QoS levels and other default settings for SLA templates. The insufficient use of formal service behaviour descriptions, varying resource demands and a choice of configuration options expected by users contribute to this issue. We present our solution of gathering monitoring data at runtime and feeding it back into the service registry to adjust descriptions and make contract template derivation a more realistic process. In addition, we show how to extend SOA building blocks such as service discoveries and SLA managers with the adjustment mechanism.

1 Introduction to SLA Management

Service Level Agreements are established between service consumers and providers and define a number of obligations and rights for both sides. Research on SLA management for web services is often focusing on SLAs with rights for consumers and obligations for providers, keeping the balance by requiring per-use or interval payment by the consumer. To attract consumers, providers need to clearly specify the pricing of service offerings.

SLAs are defined prior to service usage based on a negotiation between both sides, which can happen autonomously within any number of constraints, or can be performed interactively, even offline in some cases. Online negotiation facilities based on agents and SLA managers are increasingly available and support more dynamic service environments.

In most negotiation protocols, SLA templates or profiles are offered by the service provider to initiate the negotiation. The providers, either represented by humans or by an SLA manager service, need to know in advance of how to find a suitable ratio of payment to operational cost in order to create feasible SLA templates. This is a non-trivial task in the context of distributable services whose functional and especially non-functional properties cannot be inferred from the associated service descriptions or SLA templates.

We attempt to solve this gap by presenting a methodology for adding or adjusting the values of non-functional properties (NFPs) in service descriptions depending on the service runtime behaviour, and deriving adjusted SLA template constraints from the updated NFPs. The presented methodology separates the adjustment logics from the technical implementation needed for today's service description and SLA formats. It does not consider legal aspects and legal validity of runtime modification to SLA templates.

2 Related SLA Provisioning Approaches

There are two main reasons why monitoring data is gathered and aggregated in service hosting environments at runtime. Initially, the provider needs to infer knowledge about grey-box services with insufficient property descriptions. Throughout the service lifecycle, the existing assumptions about a service need then to be continuously validated.

When no expected behaviour is known prior to the execution, the provider can infer the knowledge incrementally through machine learning. In this case, both functional and non-functional aspects can be found out given sufficient statistical characteristics. In fact, recent work has shown that it is possible to gather information about services and service descriptions and store derived knowledge in ontologies expressed in OWL-S, e.g. [8][1]. Additionally, black and grey box testing techniques help determining the functional interfaces of services. When the expected behaviour is known, the provider can compare the observed service behaviour against the expected behaviour. The comparison is often part of the testing stage of an integrated software engineering approach, realised as continuous self-checking of systems for improving the coverage of tests for SOA-specific challenges [4]. Recently, this reason has gained momentum with approaches like models at runtime and

978-0-7695-3840-2/09 $26.00 © 2009 IEEE
DOI 10.1109/CLOUD.2009.56

model-based system management [7].

Adaptivity can also benefit from accurate NFPs, and the quality of adaptivity decisions depends on the precision of data collected from precise, ubiquitous and non-intrusive service monitoring at runtime [13]. The presented techniques help with NFP additions and adjustments in service descriptions, but are not complete regarding the handling of SLAs in contract-bound service execution.

There are several known strategies of how to estimate non-functional service properties for the creation of SLA templates. In [5], the authors propose a method and a tool suite to perform BPEL model checking at design time in order to find out the resource requirements of processes, and reason about its importance for SLAs. However, this approach is not combined with run-time monitoring. In [16], a QoS specification language and framework is presented. The major contribution of this framework to our work is the introduction of formal descriptions of SLA templates. Its shortcoming is the lack of a methodology for SLA template creation and adjustment at runtime.

Additional approaches for resource reservations in advance can be found in [11] and [10] based on fully-described component models. Planning ahead is useful and required for being able to offer guarantees, but will not suffice for dynamic service environments where resources might fail or formally specified NFPs might be influenced by previously unconsidered parameters. Therefore, the authors of [3] argue for dynamic resource provisioning to avoid SLA violation. This is a rather complementary approach to our work which could also benefit from long-term monitoring and adjustments to minimise the amount of additional provisioning after each contract negotiation. Another reason why this concept alone is not enough is that resources cannot be added indefinitely, therefore the provider needs to limit the SLA commitment level before reaching the resource limit. Another grid-oriented approach is presented in [12] which discusses transactional challenges for parallel negotiations. Feasibility checks on the side of the provider are considered necessary, but not explained further.

A closely related approach concerning SLA template adjustment focuses on the price for a service [6]. The devised architecture takes reasons for pricing differences into account. However, the approach was not implemented yet.

In summary, none of the related techniques presented above cover the entire SLA management lifecycle as they leave out the SLA template provisioning and adjustment techniques. Therefore, we want to contribute a more detailed look into the determination and dynamic adjustment of SLA templates, including price and other non-functional properties, from a service provider point of view.

3 Solution Approach: NFP Adjustments in Service Descriptions and and SLA Templates

We propose a two-step mechanism which considers the service registry adjustments of specified NFPs first and in a second phase calibrates SLA template offers accordingly. This division allows for independent use and optimisation of the algorithms.

3.1 Service Description Adjustments

NFPs of components and services can be specified formally with advanced generic languages like WSML[1] as well as with languages focusing on NFP and quality aspects like UML-OCL or CQML [16] and its derivatives μCQML, CQML+ and QML/CS. Descriptions of service NFPs are made available in service registries so they can serve as influence for service selection. In practice, NFP descriptions are often omitted, incomplete, erroneous or expressed inefficiently. Either way, monitoring services and SLA-bound service instances helps in determining the correct values for a specific environment. The underlying assumption is that similarly to predictive monitoring approaches, future values are assumed to be influenced by historic values gathered by measurement and aggregation. For each NFP, a function $predict$ works on a set of historic data nfp_i where $0 <= i <= n$ and i either refers to time-variant, event-driven measurements or to isochronous, rate-monotonic values. The effect of the measurement times on the result of prediction functions is shown in figure 1.

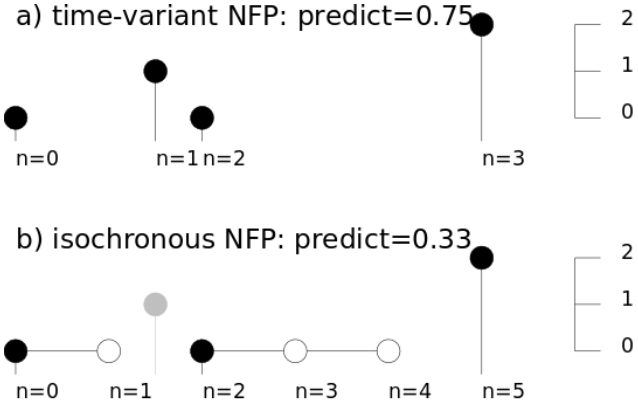

Figure 1. Dependency of NFP prediction on isochronous measurements

The prediction function returns an intermediate value nfp_{avg} based exclusively on measurements.

[1]Web Service Modelling Language: `http://www.wsmo.org/wsml/wsml-syntax`

$$nfp_{avg} = predict(nfp_0, ..., nfp_n) \quad (1)$$

A second function nfp_adjust takes previously formally specified NFPs nfp_{spec} into account and relates them to the result of $predict$ to yield the final prediction value such that:

$$nfp_{pred} = nfp\_adjust(nfp_{avg}, nfp_{spec}) \quad (2)$$

The optimal definition of both functions is subject to future research. We can assume that $predict$ calculates simple average values or smoothed averages for a certain value of n. Based on the resulting nfp_{avg}, the function nfp_adjust leans towards nfp_{spec} at the beginning to avoid high influence of outlying measurements but eventually converges towards the measured values nfp_i to account for the difference in service execution environment conditions of the provider and the conditions assumed by the service developer. If no formal NFPs are specified, this function will obviously return the result of $predict$. We will reason about candidates for these functions in the section on experimental results.

3.2 SLA Template Adjustments

SLA languages like WS-Agreement[2], WSLA or SLAng typically provide a section on constraining possible SLA negotiations and listing example templates. While a provider-side agent like an SLA manager always gets to decide on whether to accept an SLA offer or not, it is useful to infer this information with a high probability in advance. This way, negotiation roundtrips can be minimised and applications can be designed to help users negotiate an SLA. Typically, these constraints are based on business and strategic decisions which will not be exposed to service consumers. But even the sections only available to the provider in current SLA languages lack meta-constraint expressivity for automatic updates of the constraints over time. Therefore, adjustments can only happen relative to their previous values. We define another function $slat\_adjust$ which depends on the initial provider-given guarantee template $slat_{spec}$ to yield a predicted guarantee offer $slat_{pred}$ as follows:

$$slat_{pred} = slat\_adjust(slat_{spec}, nfp_{pred}, nfp_{spec}) \quad (3)$$

$$= slat_{spec} * \frac{nfp_{pred}}{nfp_{spec}} \quad (4)$$

[2]Web Services Agreement Specification: `http://www.ogf.org/documents/GFD.107.pdf`

In the case of lower-is-better guarantees like response time, from the consumer point of view, the provider will have an interest towards higher-is-better to avoid frequent SLA violations. Conversely, higher-is-better guarantees like throughput should rather be kept low in SLA adjustments. Both preferences are honoured by $slat\_adjust$.

We see the introduction of meta-constraints as an essential extension of current SLA languages to derive more business-aligned definitions of $slat\_adjust$. Such preferences already exist for client-side service selection and SLA renegotiation agents, e.g. for query preferences [9] and for semantic goal descriptions [2].

4 Experimental Results

We have written a tiny simulator in Python which in its first iteration produced simulated service response time monitoring information. While the response time NFP was initially specified by the service developer to be be $nfp_{spec} = 0.5s$ on average, the simulator was calibrated to artificially random delay around $nfp_i \cong 0.8s$. On each service invocation, the value was measured and, using a total average function, the resulting nfp_{avg} stabilised at the simulated value as expected by stochastic prediction using the knowledge about the randomness. Not having this knowledge available, nfp_{pred} still slowly converged against nfp_{avg}. The SLA template values was updated by the same relative amount due to reasons explained before. This iteration is shown in figure 2.

In a second iteration, dynamic negotiations were also simulated. SLAs were established when the price specified in the template was deemed competitive enough, for a random period of time with a depending cost and compensation amount. This time though, a long-term defect was configured to happen soon after the start which would delay the response times significantly. As can be seen in figure 3, not only were almost all of the active SLAs violated as determined by the monitoring result nfp_{avg}, but the slow convergence of guarantees in SLA templates relative to measured NFP values through nfp_adjust led to the effect that many of the newly negotiated SLAs were violated soon after their creation.

Several optimisations were tried, including experimental smoothing of the monitored values as is common with other stochastically random measurements which are hard to predict. With this technique, nfp_n depends on the previous average of $nfp_{n-1}, ..., nfp_0$ by a factor of $(1 - \alpha)$ and influences the new average by α. Eventually, a conservative choice favouring the worst response time among the total average and the exponentially smoothed average with $\alpha = 0.1$ was giving the best results. Figure 4 shows that the fast convergence leads to a quick recovery of provider compensation losses and an overall greater income after the

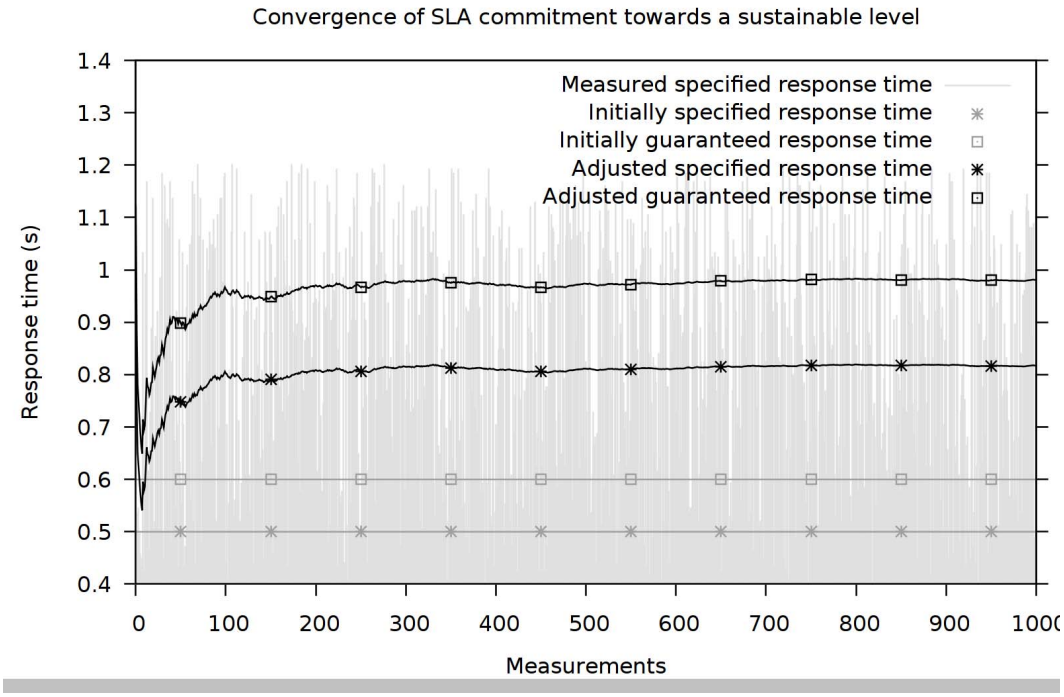

Figure 2. NFP specification converging against the average values as determined by monitoring

same period of time.

These results should not be considered a proof of optimal convergence, but they clearly show that by tuning the convergence algorithm, the SLA template offers can become more rewarding for the provider. On the other hand, the effect of potentially higher cost on the consumer can only be fully verified in a multi-provider scenario where there is some competition among providers. In the extreme case of a monopoly of only one provider, cost could be maximised and compensation minimised independently of the SLA violation frequency, whereas in a healthy competition the margins would have to be calculated as exactly as possible.

5 Extension of SOA Building Blocks

In order to demonstrate the feasibility of the two presented techniques and to verify the simulated results, an application of both techniques within existing SOA building blocks will be shown in this section. The service description updates relate to a dynamic service registry and discovery process, while the SLA template updates influence the capabilities of SLA managers.

5.1 Extension of ConQoMon for Service Description Updates

ConQoMon is a QoS- and context-aware service discovery [15]. Its registry contains WSML service descriptions. The WSML goals for queries for suitable services can be generated from the visual selection of a functional domain and non-functional requirements. Corresponding non-functional service properties are specified in the WSML files based on generic ontologies for QoS and context and can be extended with domain-specific properties. Using a matchmaking process, the most suitable service can be found by matching the required properties against the offered ones. ConQoMon is already an extension of an earlier ConQo prototype with the SLA-driven Grand SLAM monitoring framework. This makes it possible for users to look up current performance parameters of services before attempting to negotiate an SLA with them. ConQoMon stores updated NFPs reported by the monitor in its database and applies a utility function to both the original values from the WSML file and the current values to find the right value for the matchmaking process.

Using the technique presented in this paper, ConQoMon can be extended even more to incorporate the measured service properties into the service descriptions and the associated SLA templates, respectively, as shown in figure 5. We have implemented dynamic service description documents

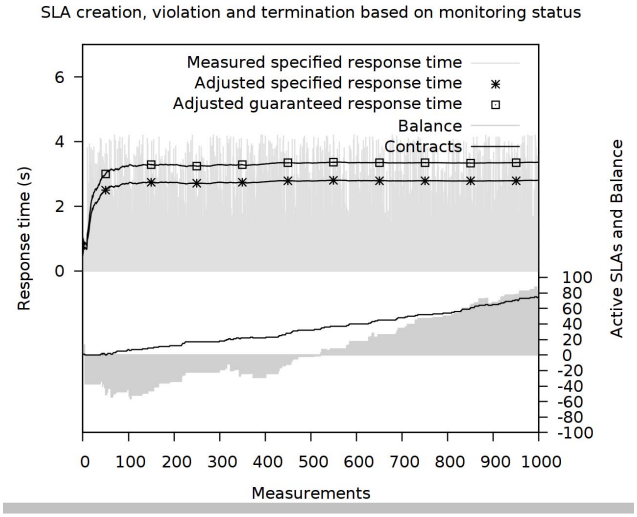

Figure 3. Dynamic SLA negotiations based on resource constraint assessment

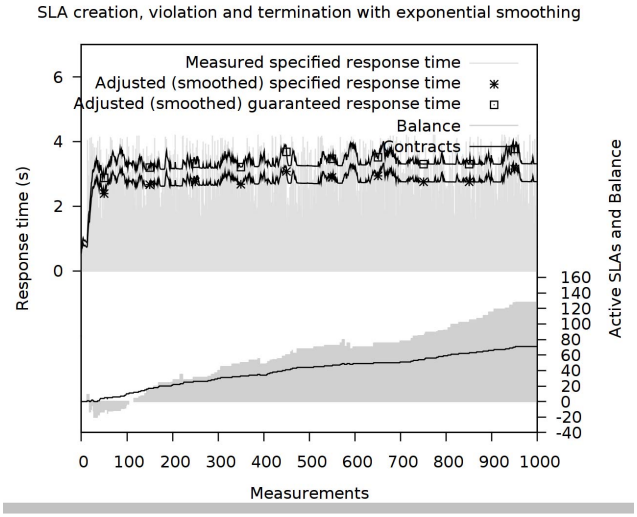

Figure 4. Dynamic SLA negotiations, variant with exponential smoothing of averaged NFP

which can now be retrieved in parallel to the unaltered documents registered at service deployment time. The NFPs in the returned WSML descriptions are updated to the values determined by nfp_adjust. For that matter, ConQoMon produces updated WSML files on request by using a rewriting mechanism based on the WSMO4J library. This technique assumes a common NFP vocabulary between the service developer, discovery users and SLA monitors. If there are multiple active SLAs associated to services and possibly multiple service instances, the NFPs are determined for all services per SLA first and then averaged depending on further SLA-specific weights. In our implementation, we assume no resource preferences based on tariffs and other SLA-specific settings and hence can assume equal weighting. Listing 1 shows an except from a WSML file including the dynamically inserted value.

Listing 1. WSML definition for a service availability characteristic

```
namespace { _"urn:...:PrinterService2.wsml#
    ",
  qos _"urn:...:QoSBase.wsml#",
  [...]
}
ontology BasicParam
  importsOntology { _"urn:...:
     PrinterContextQoSBase.wsml#" }
instance WsAvailInst1 memberOf { qos#
    Availability, qos#ServiceSpec}
  qos#unit hasValue qos#Percentage
  qos#value hasValue 0.995
```

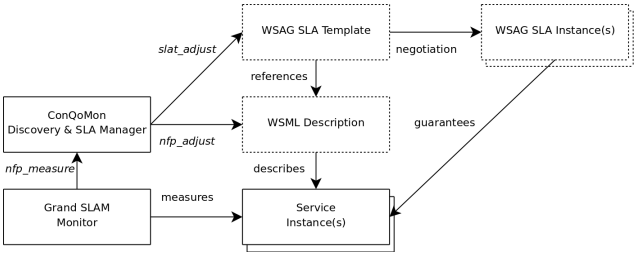

Figure 5. SLA adjustments in the presence of multiple service instances

WSML as a rather general service description language lacks a formal semantics behind NFP specifications. Therefore precise statistical distributions of monitoring information could be expressed in CQML+ in addition to the inclusion of average NFP values in WSML descriptions. We have implemented a dynamic CQML+ generator to demonstrate the smaller dependency on the usefulness of nfp_adjust. In the listing 2, an excerpt for a single NFP is shown for a randomly selected service.

Listing 2. CQML+ definition for a service throughput characteristic

```
quality_characteristic throughput {
  domain: increasing real [0..) kilobytes/s
     ;
}
quality_characteristic avg_throughput :
    throughput {
```

```
  mean;
}
quality default {
  best-effort throughput > 350.216;
  avg_throughput.mean > 437.770;
  avg_throughput.minimum > 254.013;
  avg_throughput.maximum < 536.006;
}
```

Leveraging the extensible nature of semantic service descriptions, the service ontology supports links to supplementary description files. In ConQoMon this feature is used to link WSDL files, SLA templates and CQML+ quality profiles to WSML files. In the shown architecture, the service discovery assumes the role of the SLA manager. On an implementation level, it is useful to distribute the roles onto two interacting components to maintain a separation of concerns. Therefore, the next section describes the implementation of $slat\_adjust$ with existing SLA managers.

5.2 Extension of an SLA manager for SLA Template Updates

SLA management is often encapsulated in dedicated components called SLA managers. Two representatives are the GRIA SLA Manager [3] and the TEXO SLA Manager [14]. Both already support the storage and retrieval of SLAs and SLA templates. However, neither supports integrated dynamic SLA template adjustments. Single-property guarantee offers can be updated by directly modifying SLA templates as shown in listing 3. The SLA templates which are based on WS-Agreement can either be modified in the `creationConstraints` section or in the `guaranteeTerm` section and will then be offered for negotiations from that point on, without affecting already active SLA. However, this automatic method might might still pose risks in a business context where exposed offers need to be controlled.

Listing 3. Updated WS-Agreement SLA Template in compact XML notation

```
Template templateId="SomeSLATemplate" {
 terms {
  all {
   serviceProperties {
    variableSet {
     variable name="availability" {
      location { printer#WsAvailInst1 }
     }}}
   guaranteeTerm obligated="Responder" {
    serviceScope serviceName="SomeService";
    qualifyingCondition { ... }
    serviceLevelObjective {
     kpiName { availability }
     target {
      operator { IS_ABOVE }
      value { 0.992 }
     }}
    businessValueList { ... }
   }}}}
```

We propose to extend existing SLA managers and its notions of active SLAs and SLA templates with a third document type, automatically generated SLA template proposals (SLATPs). This extension makes it possible to turn automatic SLA template adjustments into semi-automatic update suggestions which service providers can evaluate and eventually manually confirm. This concept extends current approaches such as the GRIA template administration by adding a monitoring-supported SLA template offering guidance. The SLA managers would be extended with the following operations:

- `list<SLATP> listTemplateProposals()`, to be used by an application to show automatically generated proposals to the provider;
- `bool addTemplateProposal(SLATP)`, to be used by the SLA manager extension based on an existing SLA template;
- `bool updateTemplateProposal(SLATP)`, likewise used for continuous updates by the SLA manager;
- `bool removeTemplateProposal(SLATP)`, to be used by the provider to discard unsuitable proposals; and
- `SLAT confirmTemplateProposal(SLATP)`, to eventually turn a proposal into a publicly offered SLA template.

We have implemented the logic behind this extension for the TEXO SLA Manager with an SLA template rewriting technique similar to how dynamic WSML documents are implemented. Rewriting works by replacing operators and values in WS-Agreement documents depending on the decision taken by the $slat\_adjust$ function introduced in this paper. Depending on the desired tradeoff between precision and system load, the rewriting can be scheduled to occur after every monitor measurement, after exceptional events such as service level objective violations or after periods of a fixed duration. Service providers are given the ability to modify the SLATPs through a user interface and override unsuitable decisions by $slat\_adjust$ with guarantee terms aligned with their business objectives. The user interface is shown in figure 6.

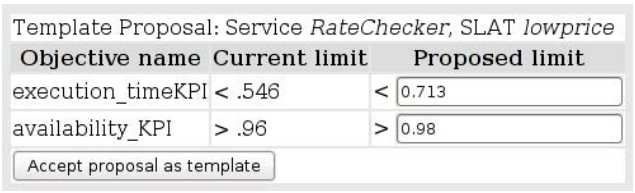

Figure 6. Semi-automatic SLA template adjustments

6 Conclusion

In fast-moving service environments with frequent SLA negotiations, service providers need to decide which guarantees can be offered for which NFPs through SLA templates. This task is non-trivial as it depends on precise and accurate NFPs in service descriptions and on business decisions. We have shown an initial attempt of how monitoring data gathered at runtime can be used to increase the accuracy of service descriptions with up-to-date NFP information and how based on these updates SLA templates can be adjusted semi-automatically aligned with business goals. Additionally, we have integrated the techniques from both steps into existing SOA platform components to produce dynamic service descriptions and SLA templates based on WSML, CQML+ and WS-Agreement. This publicly available work contributes to future SOA architectures with higher vitality and dynamics.

Acknowledgment

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

[1] M. Babik, L. Hluchy, J. Kitowski, and B. Kryza. Generating Semantic Descriptions of Web and Grid Services. In *Distributed and Parallel Systems - From Cluster to Grid Computing (Proceedings of the 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS))*, pages 83–93, 2006. Innsbruck, Austria.

[2] Y. Badr, A. Abraham, F. Biennier, and C. Grosan. Enhancing Web Service Selection by User Preferences of Non-Functional Features. In *Proceedings of the 2008 4th International Conference on Next Generation Web Services Practices (NWeSP)*, volume 00, pages 60–65, October 2008. Seoul, South Korea.

[3] M. Boniface, S. C. Phillips, A. Sanchez-Macian, , and M. Surridge. Dynamic Service Provisioning Using GRIA SLAs. In *Proceedings of NFPSLA-SOC'07*, September 2007. Vienna, Austria.

[4] G. Canfora and M. D. Penta. Testing Services and Service-Centric Systems: Challenges and Opportunities. *IT Professional*, 8(2):10–17, 2006.

[5] H. Foster, W. Emmerich, J. Kramer, J. Magee, D. Rosenblum, and S. Uchitel. Model Checking Service Compositions under Resource Constraints. In *Proceedings of ESEC/FSE*, September 2007. Dubrovnik, Croatia.

[6] P. Hasselmeyer, B. Koller, I. Kotsiopoulos, D. Kuo, and M. Parkin. Negotiating SLAs with Dynamic Pricing Policies. In *Proceedings of the SOC@Inside'07*, September 2007. Vienna, Austria.

[7] C. Hein, T. Ritter, and M. Wagner. System Monitoring using Constraint Checking as part of Model Based System Management. In *2nd International Workshop on Models@run.time*, October 2007. Nashville, Tennessee, USA.

[8] A. Heß, E. Johnston, and N. Kushmerick. Machine learning techniques for annotating semantic web services. In *Proc. Dagstuhl Seminar on Machine Learning for the Semantic Web*, 2005.

[9] M. Klein and B. König-Ries. Combining Query and Preference – An Approach to Fully Automatize Dynamic Service Binding. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, July 2004. San Diego, California, USA.

[10] M. Meyerhöfer and K. Meyer-Wegener. Estimating Non-functional Properties of Component-based Software Based on Resource Consumption. In *Electronic Notes in Theoretical Computer Science (ENTCS)*, volume 114, pages 25–45, January 2005. Barcelona, Spain.

[11] M. A. S. Netto, K. Bubendorfer, and R. Buyya. SLA-Based Advance Reservations with Flexible and Adaptive Time QoS Parameters. In *Proceedings of the 5th international conference on Service-Oriented Computing*, September 2007. Vienna, Austria.

[12] A. Pichot, P. Wieder, O. Wäldrich, and W. Ziegler. Dynamic SLA-negotiation based on WS-Agreement. Technical Report TR-0082, CoreGRID, June 2007.

[13] E. Putrycz and G. Bernard. Client Side Reconfiguration on Software Components for Load Balancing. In *Proc. International Workshop on Distributed Dynamic Multiservice Architecture, in conjunction with IEEE International Conference on Distributed Computing Systems (ICDCS)*, April 2001. Phoenix, Arizona, USA.

[14] J. Spillner, M. Winkler, S. Reichert, J. Cardoso, and A. Schill. Distributed Contracting and Monitoring in the Internet of Services. In *Proceedings of DAIS: IFIP International Conference on Distributed Applications and Interoperable Systems*, volume 5523 of *LNCS*, pages 129–142, June 2009. Lisbon, Portugal.

[15] G. Stoyanova, B. Buder, A. Strunk, and I. Braun. ConQo – A Context- and QoS-Aware Service Discovery. In *Proceedings of IADIS International Conference WWW/Internet*, October 2008. Freiburg, Germany.

[16] J. Øyvind Aagedal. *Quality of Service Support in Development of Distributed Systems*. PhD thesis, University of Oslo, Faculty of Mathematics and Natural Sciences, Department of Informatics, March 2001.