On the Automated Analysis of WS–Agreement Documents: Applications to the Processes of Creating and Monitoring Agreements

Carlos Müller
ETS. Ing. Informática, University of Seville, Spain
E-mail: cmuller@us.es

The need for Service Level Agreements (SLAs) to supervise the consumption of services is increasing in business applications and Cloud scenarios due to the penalties that can apply for violation of SLA terms. Such a need for SLAs boosts the importance and complexity of SLA lifecycle supporting systems (SLA-driven solutions). In our work we propose to improve the current support to develop SLA-driven solutions by enhancing both, the SLA specifications and automated analysis techniques. Furthermore, we have applied our proposal to creating and monitoring agreement solutions.

Keywords: service level agreements, SLA, analysis, constraint programming

1. Introduction

The need for Service Level Agreements (SLAs) to supervise the consumption of services is increasing in business applications and Cloud scenarios due to the penalties that can apply for violation of SLA terms. Such a need for SLAs boosts the importance and complexity of SLA lifecycle supporting systems, *SLA-driven solutions* from now on.

After a study of the existing literature we conclude that these SLA-driven solutions can be significantly enhanced in both: (1) the SLA specification languages, and (2) the techniques to extract useful information from the SLAs, *analysis techniques*, in advance.

On the one hand, we suggest that existing SLA specification languages can be improved by turning them into: i) domain-independent, ii) standard-recommendation-compliant, and iii) expressive-enough to ease the description of SLA terms, Service Level Objectives (SLOs), constraints, and validity periods for their comprised elements. In addition, the specification languages must define validity criteria for checking that supported SLAs satisfy some basic properties such as the consistency, and the compliance between SLA documents. Furthermore, explanations must be provided when the SLAs do not satisfy such basic properties. On the other hand, the analysis techniques that can be found in the literature include some drawbacks of emerging techniques. To overcome such drawbacks there must be developed: a) fully-functional reference implementations, b) techniques with a reuse-oriented design, c) effective extension mechanisms, and d) user-friendly interfaces.

The cornerstone of our proposal to improve the SLA specification languages has been the definition of the WS–Agreement Configurations (WSAC) that comprise the sublanguages needed to describe the different parts of Web-Services–Agreement (WS–Agreement) documents [1]. Regarding our proposal to improve the analysis techniques, the key has been the organization of such techniques in a catalogue of basic analysis operations that can be combined to support more advanced SLA-driven solutions. The applicability of our results is limited to those SLAs that can be translated to a constraint satisfaction problem (CSP) [6], that is enough to support real-world SLAs, in our experience.

2. Contributions

This section introduces the contributions described in [2] and depicted in Figure 1 regarding: SLA specification, SLA automated analysis, and SLA-driven solutions support.

Regarding the SLA specification, our first contribution (C1) is the development of a new WSAC called

AI Communications
ISSN 0921-7126, IOS Press.

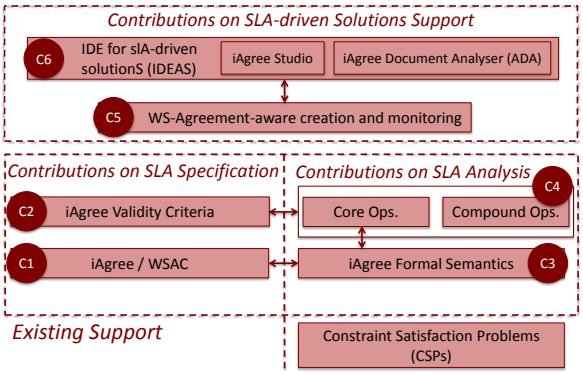

Fig. 1. Overview of the contributions

Intermediate ws–AGREEment configuration (iAgree), which allows us to specify the three kinds of WS–Agreement documents: templates, offers and agreements. The main features of iAgree are: (1) Highly WS–Agreement-compliant; (2) High and extensible expressiveness by allowing specifying expressive SLOs and constraints; (3) Domain-independent; (4) Time-aware by supporting validity periods of the different agreement elements in a very flexible way; and (5) Human-readable by means of a suitable syntax. The second contribution (C2) is the identification of a set of validity criteria for iAgree documents, which allows us to identify up to 4 kind of semantic errors: inconsistencies, dead terms, conditionally inconsistent terms, and non-compliant terms. As far as we know, we are pioneers at defining semantic errors for WSACs.

Concerning the SLA automated analysis, the first contribution (C3) consists of the definition of a formal semantics for iAgree. This provides the iAgree constructions with precise meaning, and eases the automated extraction of information from documents written in iAgree. In particular, iAgree semantics is based on CSPs. Next, relying on iAgree semantics, we have developed an approach to automatically check and explain violations of iAgree validity criteria [4], organized in a catalogue of four *core operations*, and some additional analysis operations (C4). A CSP-based reference implementation has been developed to support the execution of all the operations in SLA-driven solutions.

With respect to the SLA-driven solutions support, our first contribution (C5) aims at extending both, SALMon and WSAG4J, two well known SLA-driven solutions, to become them WS–Agreement and iAgree compliant by using our analysis operation catalogue. SALMon is a monitoring platform which main goal has been to gather the quality of service of web services and check simple SLA conditions. We extended SALMon jointly with its creators [3] in order to support: (1) the analysis of WS–Agreement-compliant agreements with expressive SLOs, and (2) checking and explaining agreement violations. In turn, WSAG4J tool provides an implementation of the WS–Agreement protocol. We extended WSAG4J jointly with its creators [5] in order to endow it with analysis facilities in such a way that templates, offers and agreements can be validated before, during and after the agreement creation.

The second contribution aims at providing support for both human and software clients that are involved in the development of SLA-driven solutions (C6). In this case, we provide a novel Integrated Development Environment for slA-driven Solutions (IDEAS), since to the best of our knowledge there is not exist any similar one. IDEAS provides a publicly-available user-friendly front-end which makes possible: (1) to edit documents assuring they are valid, and (2) to analyse some properties appealing for final and technical users.

Acknowledgements

I would like to thank A. Ruiz-Cortés, and M. Resinas for the supervision and support of this work that has been partially supported by the Spanish and the Andalusian R&D&I programmes (grants IPT-2012-0890-390000, IPT-2013-0890-3, P12–TIC–1867, TIN-2012–32273, TIC–5906, TIN2009–07366, P07–TIC–2533).

References

[1] Andrieux et al. Web Services Agreement Specification (WS-Agreement) (v. gfd-r.192), 2011. OGF - Grid Resource Allocation Agreement Protocol WG.

[2] C. Müller. *On the Automated Analysis of WS-Agreement Documents. Applications to the Processes of Creating and Monitoring Agreements.* International dissertation, 2013, http://www.isa.us.es/sites/default/files/muller-Phd-PTB.pdf.

[3] C. Müller, M. Oriol, X. Franch, J. Marco, M. Resinas, A. Ruiz-Cortés, and M. Rodríguez. Comprehensive Explanation of SLA Violations at Runtime. *IEEE Trans. on Services Computing*, 2013.

[4] C. Müller, M. Resinas, and A. Ruiz-Cortés. Automated Analysis of Conflicts in WS–Agreement Documents. *IEEE Trans. on Services Computing*, 2013.

[5] O. Wäldrich, H. Rasheed, and W. Ziegler. WS–Agreement for Java Framework (WSAG4J) by Fraunhofer SCAI Institute, and members of GRAAP-WG of the Open Grid Forum. `http://packcs-e0.scai.fraunhofer.de/wsag4j/`.

[6] E. Tsang. *Foundations of Constraint Satisfaction*. A. Press, 1995.