iAgree Studio: A Platform to Edit and Validate WS–Agreement documents[⋆]

C. Müller, A.M. Gutiérrez, M. Resinas, P. Fernández, and A. Ruiz–Cortés

University of Seville, LSI
ISA research group, `http://www.isa.us.es/`, Seville (Spain)
`{cmuller,amgutierrez,resinas,pablofm,aruiz}@us.es`

Abstract. The widespread use of SLA-regulated Cloud services, in which the violation of SLA terms may imply a penalty for the parties, have increased the importance and complexity of systems supporting the SLA lifecycle. Although these systems can be very different from each other, ranging from service monitoring platforms to auto-scaling solutions according to SLAs, they all share the need of having machine-processable and semantically valid SLAs. In this paper we present iAgree studio, the first application, up to our knowledge, that is able to edit and semantically validate agreement documents that are compliant with the WS–Agreement specification by checking properties such as its consistency, and the compliance between templates and agreement offers. In addition, it reports explanations when documents are not valid. Moreover, it allows users to combine the validation and explanation operations by means of a scenarios developer.

1 Overview and Motivation

SLAs are widely used nowadays as a means to regulate the terms and conditions under which a service is provided. As the use of SLAs in Cloud services and applications in which the violation of SLA terms may imply a penalty for the parties increases, the complexity and demand of systems supporting the SLA lifecycle also increases. These systems include service monitoring platforms that use SLAs to decide which service metrics should be monitored, auto-scaling solutions that automates the provisioning or deprovisioning of resources according to the SLA, and billing components that calculate the penalties incurred during the use of a service, amongst others. Although very different from each other, all of these systems require having semantically valid SLAs (i.e., without semantic errors) and defined in a machine processable manner.

WS–Agreement [1] is arguably the most widespread recommendation for defining machine processable SLAs. It specifies a template-based agreement creation protocol and an XML Schema that defines the basic structure of an SLA and the other documents used in the agreement creation protocol like agreement templates and agreement offers. However, WS–Agreement leaves open how the different elements of a WS–Agreement document such as a Service Level Objective (SLO) must be specified.

[⋆] This work was partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2009–07366 (SETI), TIN2012–32273 (TAPAS), TIC–5906 (THEOS)).

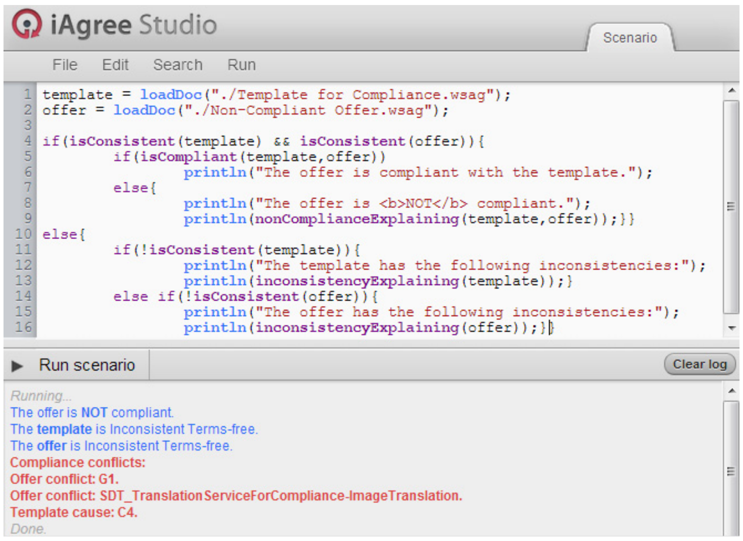
```
Template AmazonS3 version 1.0
  Provider Amazon as Responder;

AgreementTerms
  Service AWS-S3 available at. aws.amazon.com/s3    //Service reference in iAgree
    Global description:  //Service description term in iAgree
      Interface;                    //either SOAP or REST
      RRS = False;                  //Reduced Redundancy Storage (RRS)
      StorageSize;                  //StorageSize in TB
      FirstProject;                 //Denotes if it is the first customer project
      TotalPrice, StoragePrice, SupportPlanPrice;

  Monitorable Properties  //Service properties in iAgree
    global:
      MUP;   //Monthly Uptime Percentage, a kind of AmazonS3 service availability
      TransferredGb; ResponseTime; ReadRequests, WriteRequests;
      OnlineReportingSupport, PhoneSupport;  //Customer support facilities
      TurnAroundTime;                        //Minutes to solve problems

  Guarantee Terms
    G1: Provider guarantees MUP >= 99.9;
    G2: Consumer guarantees TransferredGb < StorageSize * 100
                            AND ReadRequests > WriteRequests;
    G3: Provider guarantees ResponseTime < 1000; onlyIf (Interface = SOAP);
    G4: Provider guarantees ResponseTime < 700; onlyIf (Interface = REST);
    G5: One or More between:
      G5.1: Provider guarantees OnlineReportingSupport = true;
      G5.2: Provider guarantees TurnAroundTime = 15;
      G5.3: Provider guarantees PhoneSupport = true;

Creation Constraints:
  C1: StoragePrice = 0.05 * StorageSize; onlyIf RRS = true;
  C2: StoragePrice = 0.12 * StorageSize; onlyIf RRS = false;
  C3: StorageSize <= 5000 TB;
  C4: TotalPrice = StoragePrice + SupportPlanPrice;
      onlyIf (FirstProject = false or StorageSize > 5);
```

Fig. 1. Template of AmazonS3 service scenario in iAgree

iAgree [4] is a fully-fledged WS–Agreement-compliant language that completes the WS–Agreement schema with a set of languages to describe all WS–Agreement elements. Figure 1 shows an iAgree template inspired in the Amazon Simple Storage Service (AmazonS3) including: terms to describe the service (see `service AWS-S3`, and `Monitorable Properties`) and guarantees (see `G1-G5`), terms compositors to combine the terms (see `G5`), and creation constraints (see `C1-C4`). Moreover, iAgree supports expressive arithmetic-logic expressions within the service level objectives (SLOs) (see `G2`), qualifying conditions (QCs) of conditional terms (see `G3-G4`), and creation constraints (CCs) (see `C2`). In addition, an advantage of iAgree is that its validity criteria has been extensively researched [2,3,4] and algorithms for checking and explaining the validity of iAgree documents have also been developed.

Based on those results, in this paper we present *iAgree studio*[1], a web application to edit and validate iAgree documents. In particular, it supports the kinds of conflicts between terms and creation constraints presented in [3,4], and the non-compliance situations between templates and agreement offers exposed in [2].

[1] Available at `www.isa.us.es/iagreestudio/`, including a screencast.



```
template = loadDoc("./Template for Compliance.wsag");
offer = loadDoc("./Non-Compliant Offer.wsag");

if(isConsistent(template) && isConsistent(offer)){
        if(isCompliant(template,offer))
                println("The offer is compliant with the template.");
        else{
                println("The offer is <b>NOT</b> compliant.");
                println(nonComplianceExplaining(template,offer));}}
else{
        if(!isConsistent(template)){
                println("The template has the following inconsistencies:");
                println(inconsistencyExplaining(template));}
        else if(!isConsistent(offer)){
                println("The offer has the following inconsistencies:");
                println(inconsistencyExplaining(offer));}}
```

```
Running...
The offer is NOT compliant.
The template is Inconsistent Terms-free.
The offer is Inconsistent Terms-free.
Compliance conflicts:
Offer conflict: G1.
Offer conflict: SDT_TranslationServiceForCompliance-ImageTranslation.
Template cause: C4.
Done.
```

Fig. 2. Scenario to check and explain compliance problems.

2 Novelties and Functions

The novelty of iAgree Studio is given by the following features[2]:

High WS–Agreement compliance. iAgree studio supports to edit and validate WS–Agreement documents with expressive terms including arithmetic-logic expressions relating several service properties inside the SLOs, QCs, and CCs, and supporting terms compositors defining agreement variants inside an agreement. Other WS–Agreement-based solutions studied in [4] do not support these agreement elements that are in the specification limiting their usefulness in real scenarios in which many of these elements are commonly used.

Document validation. iAgree studio is able to validate iAgree documents by checking that they do not contain semantic errors, supporting the kinds of conflicts between terms and creation constraints presented in [3,4], and the non-compliance situations between templates and agreement offers exposed in [2]. Depending on the kind of document, the validation comprise different properties.

Semantic errors explanations. iAgree studio provides an explanation report after the documents validations when semantic errors are detected. Such reports include the terms and creation constraints that are involved in the detected semantic error. For instance, a contradiction between terms, or a non-compliance between offers and template terms that make them non-compliant.

Scenarios developer. iAgree studio incorporates a *Scenarios developer* that allows users to combine the validation operations and explanation reports to obtain advanced and customisable validation scenarios. For instance, an interesting scenario

[2] Note that iAgree studio is an ongoing work and it will be extended in a nearby future with more features.

may be to check the validity of agreement offers and templates before checking the compliance between them. Such scenario is included in Figure 2 for a specific pair of documents including the explanation reports if semantic errors are detected.

In addition, iAgree studio has been tested by our M.Sc students in an SLA learning course and they suggested a number of user-friendly facilities that have been incorporated in current iAgree studio version such as: menus structure organised as in google docs, coloured syntax to highlight iAgree keywords, undo-redo functions, documents can be downloaded in iAgree or a serialised XML-based syntax, several samples presented in [3] are preloaded to try the iAgree studio functionality, etc.

3 Internal structure

The automated checking and explanation for semantic errors included within iAgree documents is performed by a Constraint Satisfaction Problems [5] (CSP)-based technique implemented within an iAgree Document Analyser (ADA) (cf. Figure 3). Such an automated technique helps the parties involved in achieving an agreement during the whole SLA-lifecycle as follows: when the documents are edited their validity can be assured because the possible semantic errors are reported in the iAgree studio to be solved; when the documents are interchanged at negotiation time, the validity of documents is also assured and the compliance between them can also be checked to ensure their compliance; afterwards, the deployment of valid SLAs is granted.

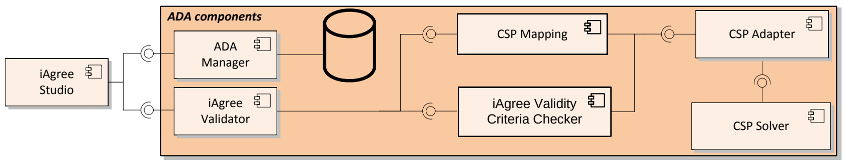


Fig. 3. Structure of our approach.

References

1. Andrieux et al.: Web Services Agreement Specification (WS-Agreement) (v. gfd-r.192) (2011), OGF - Grid Resource Allocation Agreement Protocol WG
2. Müller, C., Resinas, M., Ruiz-Cortés, A.: Explaining the Non-Compliance between Templates and Agreement Offers in WS-Agreement*. In: Proc. of the 7^{th} ICSOC'09. LNCS, vol. 5900
3. Müller, C., Resinas, M., Ruiz-Cortés, A.: Automated Analysis of Conflicts in WS–Agreement Documents. IEEE Transactions on Services Computing (2013), `http://dx.doi.org/10.1109/TSC.2013.9`
4. Müller, C.: On the Automated Analysis of WS-Agreement Documents. Applications to the Processes of Creating and Monitoring Agreements. International dissertation, Universidad de Sevilla (2013), `http://www.isa.us.es/sites/default/files/muller-Phd-PTB.pdf`
5. Tsang, E.: Foundations of Constraint Satisfaction. A. Press (1995)