A Framework to Analyse WS–Agreement Documents*

Carlos Müller, Manuel Resinas, and Antonio Ruiz-Cortés

Dpto. Lenguajes y Sistemas Informáticos
ETS. Ingeniería Informática - Universidad de Sevilla
41012 Sevilla (Spain - España)
{cmuller, resinas, aruiz}@us.es

Abstract. The WS–Agreement recommendation has gained a wide acceptance in the grid and web services communities as a means to establish service level agreements between service providers and consumers. Although there are some implementations of the WS–Agreement protocol such as WSAG4J, the lack of user–friendly tools for the edition and analysis of WS–Agreement documents is still a problem. In this paper, we present an Agreement Document Analysis (ADA) framework for the automated analysis of WS–Agreement documents. ADA contributes to achieve consistent and compliant WS–Agreement documents by supporting the detection and explanation of conflicts amongst the terms of the documents. Moreover, ADA has been designed to support extensions in aspects such as the analysis operations, the accepted agreement document formats, and even the paradigm to automate the analysis operations. In addition, analysis engines derived from ADA can be deployed as a Java library, an OSGi bundle and/or a WSDL web Service. Finally, the possible synergy between ADA and WSAG4J is also revised.

1 Introduction

WS–Agreement [1] is a proposed recommendation of the Open Grid Forum (OGF) that has gained a wide acceptance in the grid and web services community as a means for establishing service level agreements (SLAs) between service providers and consumers. WS–Agreement provides an XML schema for defining SLAs and a protocol for creating them based on templates and agreement offers. The structure of a SLA in WS–Agreement is a document that consists of a set of terms that include information about functional features, non-functional guarantees and any other terms with information relevant to the agreement. Due to its general purpose, WS–Agreement leaves open the concrete specification language of several SLA elements such as those regarding to domain–specific terms and

* This work has been partially supported by the European Commission (FEDER) and the Spanish Government under the CICYT project SETI (TIN2009–07366), and by the Andalusian Government under the project P07–TIC–2533. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

to the language to express conditions and guarantees in the agreement. Thus, WS–Agreement can be seen as a framework for writing and creating SLAs.

Although there are some implementations of the WS–Agreement protocol [2], the lack of user–friendly tools for the edition and analysis of WS–Agreement documents is still a problem. In particular, analysing WS–Agreement documents is an error–prone and tedious task, and it is infeasible to do manually with large documents. Therefore, the automated analysis of WS–Agreement documents, i.e., extracting information from WS–Agreement documents using automated mechanisms [19], is a very appealing feature for those implementations.

A number of operations of analysis can be defined for WS–Agreement documents. In this paper, we focus on automated analysis operations that grant the creation of conflict–free WS–Agreement documents. There are many circumstances under which the WS–Agreement document terms may conflict with each other during an SLA lifecycle. However, we consider two kinds of conflicts: (1) inconsistency conflicts, when contradictory agreement elements are included in the same WS–Agreement document, either a template or an agreement offer; and (2) non-compliance conflicts, when some agreement offer elements are in contradiction with some template elements. There is a conflict when two or more elements of the involved documents state opposite assertions and, hence, they cannot be fulfilled at the same time.

There are several proposals that automatically check SLAs for inconsistency or non-compliance [3–14]. However, none of them provide a tool or framework with support for the detection and the explanation of the causes of both kinds of conflicts. This makes these proposals of limited utility because identifying and solving conflicts by hand may become a daunting task, specially if the SLA is large or it offers many alternatives and conditions.

Solution overview and contributions. In this paper, we present an Agreement Document Analysis (ADA[1]) framework for the automated analysis of WS–Agreement documents, including the detection and explanation of inconsistency and non-compliance conflicts. ADA contributes to achieve consistent and compliant WS–Agreement documents, by supporting all kinds of conflicts previously identified by us [16,17]. Moreover, ADA has been designed to support extensions in aspects such as the analysis operations, the accepted agreement document formats, and even the paradigm to automate the analysis operations. Furthermore, third–parties tools and frameworks such as WSAG4J[2] can interact with ADA by means of several interfaces. Java, OSGi[3], and WSDL[4] interfaces are provided. We show in this paper how it is used through a web service interface and we

[1] `http://www.isa.us.es/ada`

[2] WS–Agreement framework for Java (WSAG4J). `http://packcs-e0.scai.fraunhofer.de/wsag4j/index.html`

[3] OSGi Service Platform Core Specification, by the OSGi Alliance `http://www.osgi.org/download/r4v42/r4.core.pdf`

[4] Web Service Description Language (WSDL), by the World Wide Web Consortium (W3C). `http://www.w3.org/TR/2007/REC-wsdl20-20070626/`

include some use cases using a web–based agreement document editor developed by us.

This paper is structured as follows. Section 2 presents the analysis operations for WS–Agreement documents previously studied by us. Section 3 explains ADA features, extension points and the synergy between ADA and WSAG4J. Section 4 exposes the alternative techniques available to interact with ADA, giving details for the interface provided by ADA and its deployment as a web service. Section 5 shows ADA in action by means of a developed rich internet application with some use cases. Finally, Section 6 details our conclusions and future work.

2 Analysing WS–Agreement documents

During the last years we have been studying techniques to perform an automated analysis of expressive WS–Agreement documents to make the agreement creation process easier for involved parties [15–17]. In ICSOC'07 [15] we proposed a temporal domain specific language (DSL) to increase the temporal-awareness of WS–Agreement specification. Such temporal DSL, allows expressive periodical/non-periodical and disjoint/non-disjoint validity periods to the terms of SLAs. Later on ICSOC'08 [16] and ICSOC'09 [17] we dealt with the automated analysis of SLAs proposing an automated mechanism to explain the inconsistency of a WS–Agreement document and the non-compliance between an agreement offer and an agreement template, respectively. Afore mentioned papers [16, 17] were inspired by previous papers [3, 20] which focus on checking whether an SLA is compliant with another one but without providing any explanation for the non-compliance, if any.

In the following Sections we explain in more details the automated analysis operations for WS–Agreement documents studied by us in previous works without considering the temporal DSL proposed in [15] because it regards with an expressiveness improvement instead of with the automated analysis of WS–Agreement documents. To make them easier to understand, we use an example based on a scenario in which a provider offers translation services including non-functional properties such as: *cost*, *size of the input text to translate* (size) -in number of words-, and *demanded time* (time) -in days-.

2.1 Consistency and Compliance Checking

The consistency and compliance checking were the two first analysis operations studied by us in [3, 20]. However, those works deal with our own SLA specification language. Afterwards, in [16, 17], we adapted the operations to check the consistency of any WS–Agreement document and to check the compliance between agreement offers and templates.

On the one hand, there is an inconsistency when two or more elements of the document state opposite assertions and, hence, they cannot be fulfilled at the same time. This means that an agreement with inconsistent terms can never be completely fulfilled because one of their terms will never be. For instance, in

the translation service scenario, assuming that: size ranges between $[100, 1000]$, time ranges between $[1, 5]$ and cost = size/time, if the document includes a term obligating to pay a cost of 19, at a first sight the document seems consistent. However, the lowest valid value for cost is 100/5=20, so this term cannot be fulfilled.

On the other hand, an agreement offer is not compliant with a template (i.e. there is a non-compliance conflict), if any agreement offer element state assertions which are in contradiction with the creation constraints in the template. Furthermore, in [17] we extend this WS–Agreement compliance definition considering all offer and template elements. For instance, in the previous translation service scenario, the cost definition (cost = size/time) could be included in the template as a creation constraint and if the agreement offer obligates to pay a cost of 19, there is a non-compliance between documents as occurs with the previous inconsistency sample.

In both cases, our approach to check consistency and compliance conflicts involves mapping the WS–Agreement documents into a constraint satisfaction problem (CSP) [19] and then using a CSP solver to check those conflicts.

2.2 Inconsistency and Non-Compliance Explanation

The inconsistency and non-compliance explanation analysis operations were studied by us in [16, 17].

The explanation is performed by identifying the set of contradictory elements for each conflict to be solved. Desktop applications were developed as proof-of-concept for both operations in which only a minimal subset of conflicting terms is showed to users to make it easier to solve the conflict. For instance, in the previous scenario assuming the same ranges for variables, the explanation for the conflict would be the cost definition cost = size/time, and the term obligating to pay a cost of 19. In addition, the information of which document includes each conflicting element is provided in the non-compliance explanation. Providing an explanation of why there is a conflict is very appealing from a practical point of view since identifying and solving conflicts by hand may become a daunting task, specially when the document is large or it offers many alternatives and conditions.

3 ADA Framework

ADA is an *Agreement Document Analysis* framework with the aim of extracting useful information from agreement documents. ADA is developed in Java mainly based on our previous theoretical works on applying the *constraint satisfaction problem* (CSP) [19] paradigm to the automated explanation of WS–Agreement document inconsistency and non-compliance situations [16, 17], mentioned in the previous section. The main goal of ADA is to provide WS–Agreement practitioners not only with the automated capabilities of checking consistency and compliance properties of agreement offers and templates but also with the corresponding explanations if some conflicts are identified once checks are performed.

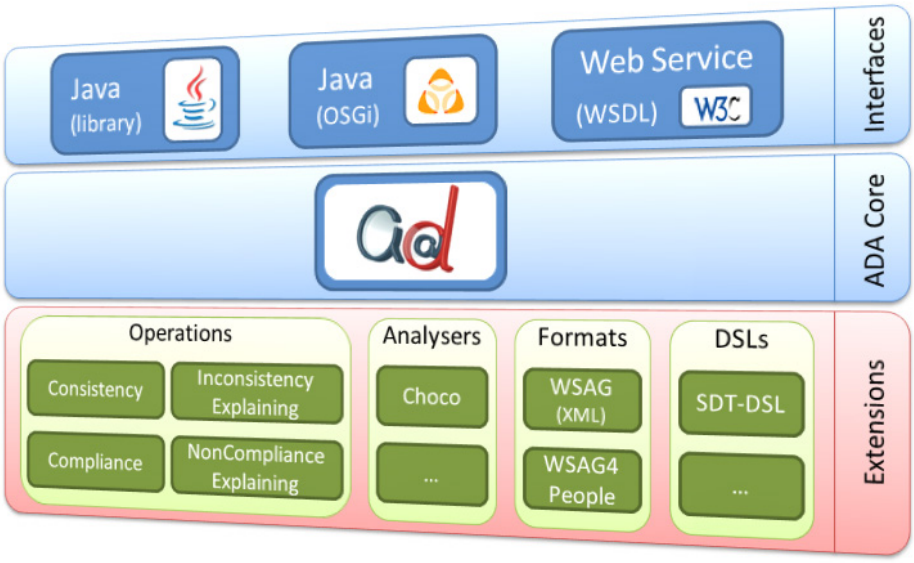

Fig. 1. Plug-in oriented architecture of the ADA framework

3.1 ADA Structure

Foreseeing the need for more analysis operations, we have designed a *three–layered plug–in oriented architecture* for ADA so it could be easily extended in the future with new capabilities (see Fig. 1). From our experience on agreement document analysis [3,16,17], we have identified the common features of different analysis proposals. Those features and the different extension points are part of the ADA framework core, the main layer of ADA and included in the middle of Fig. 1. The interfaces layer depicted at the top of Fig. 1 describes three alternative interfaces to grant the ADA access for multiple types of clients and Section 4 describe it in more details. The bottom layer of Fig. 1 shows the kind of extensions provided by ADA:

- *Operations*: currently, ADA includes as analysis operations those mentioned in Section 2, but it allows the inclusion of new analysis operations by developing concrete plug–ins. Those plug–ins are extremely easy to include in ADA if the new analysis operations can be interpreted as a constraint satisfaction problems. However, other kinds of analysis operations could be also added.
- *Analysers*: currently, ADA uses the open source Java library for constraint satisfaction problems Choco[5] to develop the analysis operations, although other solvers, not necessarily CSP–oriented, could be used in the future.

[5] Choco solver `http://www.emn.fr/z-info/choco-solver/`

- *Formats of inputted agreement documents*: currently, ADA support the analysis of inputted WS–Agreement documents in both: the XML language of the recommendation [1] or WSAg4People [18][6], a plain–text language equivalent to the XML schema but easier to handle by humans. However, support for more agreement formats can be added in future.
- *DSLs*: In order to achieve our goals on the analysis of WS–Agreement documents, some DSLs must be set as the concrete choice of some extension points established by WS–Agreement recommendation. Those extension points affects to the terms that describe the service features and properties, the guarantees and other constraints. More information about this DSLs considerations can be obtained in [18] and inside *related publications* of *documentation* section at `http://www.isa.us.es/ada`.

3.2 ADA–WSAG4J Synergy

WSAG4J framework developed by Wäldrich et al., has a different purpose than ADA. WSAG4J focuses on implementing the WS–Agreement protocol and providing support for monitoring the agreements, whereas ADA focuses on the automated analysis of WS–Agreement documents, i.e., it focuses on extracting useful information from WS-Agreement documents using automated mechanisms [19]. The unique common feature between WSAG4J and ADA is the validation of agreement offers based on template creation constraints, which correspond to the analysis operation of compliance checking in ADA. However, besides creation constraints, ADA checks the compliance amongst offer terms and template terms and supports both syntactic and semantic checking [17]. Furthermore, ADA includes the consistency checking, and it provides explanations for the consistency and non-compliance conflicts when they are detected.

Therefore, ADA can be used to extend the capabilities of WSAG4J to analyse automatically WS–Agreement documents. In particular, it would benefit from ADA's consistency and compliance checking and explanation that we have described in Section 2 as well as additional analysis operations that are being developed in ADA. In this sense, we are currently in contact with GRAAP-WG[7] members, and WS–Agreement/WSAG4J responsibles, to support the ADA analysis capabilities from WSAG4J framework. For instance, in a negotiation process to create an agreement as we detail as follows: The negotiation process may be initialised by party that publish their capabilities and demands in a template. Then, several agreement offers based on such template may be sent to the template owner. At this moment, is very appealing for the template owner to analyse in an automated way if the incoming agreement offers are compliant with the template terms or not. In the latter case, it is also of interest for the agreement initiator to know the concrete conflicting offer terms to react by preparing a counter-offer, or even by changing the template demands, if needed. The next

[6] There are examples and a more thorough description of WSAg4People available at `http://www.isa.us.es/ada`

[7] Grid Resource Allocation Agreement Protocol Working Group, (Object Grid Forum)

section details how this interaction between ADA and external tools and frameworks may take place.

4 Interacting with ADA

In order to get a complete analysis support during the whole SLA lifecycle, currently we offer three ADA facades to allow multiple types of clients. Thus, as interfaces of Fig. 1 denotes, developers can use ADA through a conventional Java library; or as a Java OSGi bundle, for example for its integration in the *Eclipse*[8] platform; or as a web service through a WSDL automatically generated from the OSGi bundle and deployed into an OSGi web container as Fig. 2 depicts. Independently of the technique used to interact with ADA, the following interface is provided:

```
ADA interface summary {

  checkDocumentConsistency (WS-Ag doc): boolean
  // it returns true if the inputted document
  // is consistent and false otherwise.

  explainInconsistencies (WS-Ag doc): Explanation set
  // it returns the conflicting terms, if any,
  // of inputted document.

  hasWarnings (WS-Ag doc): boolean
  // it returns true if the inputted document
  // includes warnings -conflicts which does not
  // make the document inconsistent -.

  explainWarnings (WS-Ag doc): Explanation set
  // it returns the warnings, if any, of inputted document,
  // and false otherwise.

  isCompliant (WS-Ag template, WS-Ag offer): boolean
  // it returns true if the inputted documents are compliant.

  explainNonCompliance (WS-Ag template,
                        WS-Ag offer): Explanation set
  // if there are conflicting terms, it returns the
  // conflicting offer and template terms relating them.

  xmlToWSAg4People (XML WS-Ag doc): WSAg4People doc
  // it translates the inputted XML WS-Agreement document
  // into the equivalent WSAg4People document.

  wsag4PeopleToXML (WSAg4People doc): XML WS-Ag doc
  // it translates the inputted WSAg4People document into
  // the equivalent XML WS-Agreement document.
}

About parameters:
  * WS-Ag doc. Either an agreement offer or a template
    in XML language or in WSAg4People.

  * WS-Ag template, WS-Ag offer. Either in XML language
    or in WSAg4People.

  * XML WS-Ag doc. Either an agreement offer or a template
    in XML language of WS-Agreement
```

[8] The Eclipse Foundation `http://www.eclipse.org`

```
* WSAg4People doc. Either an agreement offer or a
  template in WSAg4People language

* Explanation set. It is a set of conflicting terms.
  The relation between them is provided if they are
  non-compliant conflicting terms.
```

1.1. ADA interface summary

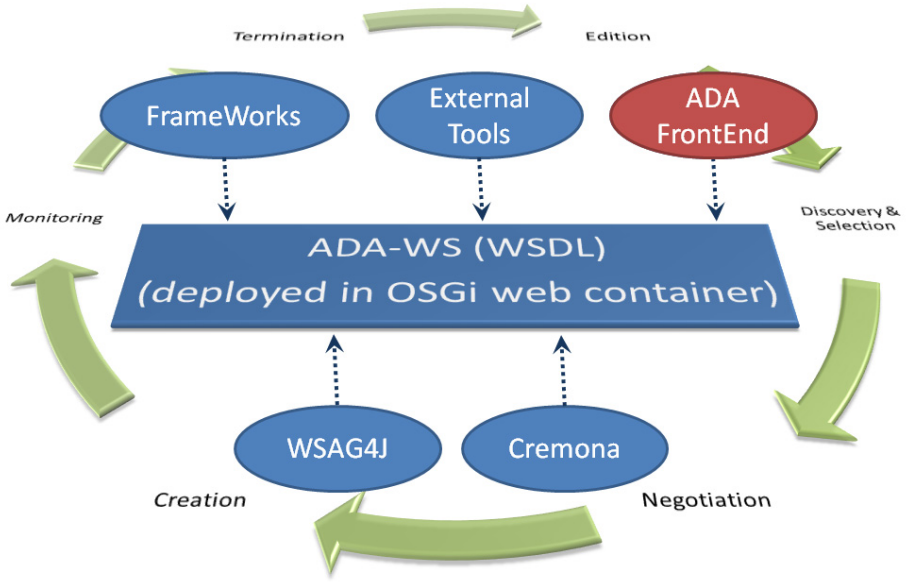

Fig. 2. ADA framework deployed as a web service into an OSGi web container

5 ADA in Action

In order to provide the community of WS–Agreement practitioners with an user–friendly environment for the edition and analysis of WS–Agreement documents and get feedback for our research, we have developed a rich user interface (ADA–FrontEnd) that can be accessible via web using the *JavaFX*[9] technology at the ADA website (`http://www.isa.us.es/ada`), including demonstration videos in English and Spanish.

Some of the most relevant features of the rich user interface are the following, where the number in parenthesis refers to the corresponding part of Fig. 3:

[9] Oracle Corporation. `http://javafx.com`

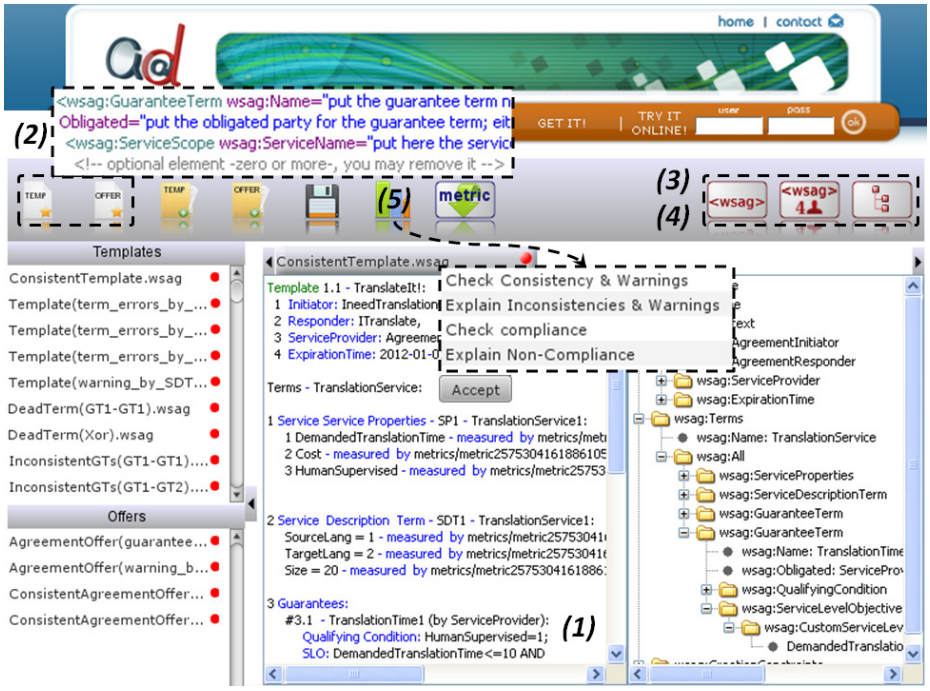

Fig. 3. Screenshot of the rich user interface (ADA-FrontEnd) to try ADA framework

(1) *syntax highlighting* for XML and WSAg4People languages; (2) *skeletons* for the creation of agreement documents, specially developed for users not familiar with the WS–Agreement complexity; (3) *multiple views* of agreement documents, including an XML view, a WSAg4People view and a *tree* view, that can be easily switched by the user using the button panel in (4); (5) a *wizard* to launch analysis operations and several document samples preloaded, in order to reduce the learning curve.

The developed editor is only a possible front end of ADA framework but it allows to show how ADA may help to obtain conflict-free WS-Agreement documents. At following we detail some use cases of conflict explaining in the afore mentioned translation service providing scenario. In such scenario, a translation service provider would use our editor to get conflict-free agreement documents.

5.1 Getting Consistent WS-Agreement documents

Afore mentioned translation service could be described by the provider in a template with expressive service level objectives (SLO) as Document 1.2[10] depicts.

[10] The document is shown in the cited human-readable WSAg4People notation [18] but some of the elements has been summarised.

```
Template 1 – Translate it!
  1 Initiator: IneedTranslation Corp.,
  2 ServiceProvider: AgreementResponder

Service Description Terms:
  1 SDT1 – TranslationService1
    SourceLang = 1, //Source language to be translated, ''1=English''
    TargetLang = 2, //Target language to traslate, ''2=Spanish''

Service Properties:
 1 SP1 – TranslationService1
  1 DemandedTranslationTime – measured By Time,        //User demanded time
  2 InputErrors – measured By Percentage,       //% of typos in source text
  3 Cost – measured By MediumInteger,                  //Cost for the service
  4 HumanSupervised – measured By BinaryInteger,//translation supervision
  5 Size – measured by MediumInteger;         //Size of source text in pages

Guarantees:
  1 TranslationTime1 (By ServiceProvider):
    Qualifying Condition: HumanSupervised=1;
    SLO: DemandedTranslationTime<=10;

  2 TranslationTime2 (By ServiceProvider):
    Qualifying Condition: HumanSupervised=2;
    SLO: DemandedTranslationTime<=100;

  3 InputErrorsGT1 (By ServiceConsumer):
    SLO: InputErrors<=1;

  4 InputErrorsGT2 (By ServiceConsumer):
    SLO: InputErrors>1;

  5 MinimunTranslationTime (By ServiceProvider):
    SLO: DemandedTranslationTime>0;

Creation Constraints:
 Items:
  1  SourceLang: Value Of SourceLang Is xs:integer [1,3],
  2  TargetLang: Value Of TargetLang Is xs:integer [1,3],
  3  Size: Value Of Size Is xs:integer [1,50],
  4  Cost: Value Of Cost Is xs:integer [1,100],
  5  HumanSupervised: Value Of HumanSupervised Is xs:integer [1,2],

 Constraints:
  1 HumanSupervisedCreationConstraint:
    HumanSupervised=1;

  2 Implication1CreationConstraint:
    HumanSupervised=1 IMPLIES Size>20;

  3 Implication2CreationConstraint:
    HumanSupervised=1 AND Size<20;
```

1.2. Template of translation service scenario

Template 1.2 has inconsistency conflicts included by provider mistakes. Therefore, the provider would be interested in analysing the document before publishing the template to avoid problems. The first help provided by the editor is the log console shown in Figure 4 with a summary of the explanation. Then, to make easier for the users to find the conflicts, as Figure 5 denotes, they are high-

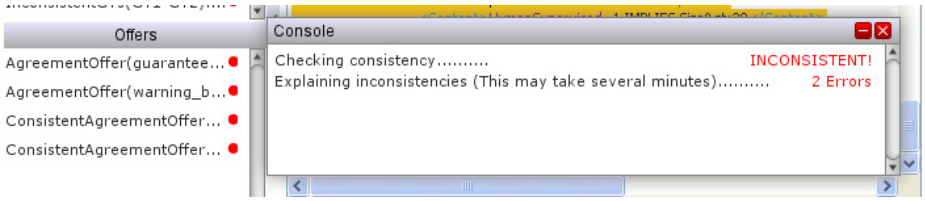

Fig. 4. log console screenshot to trace the inconsistency explaining inconsistencies

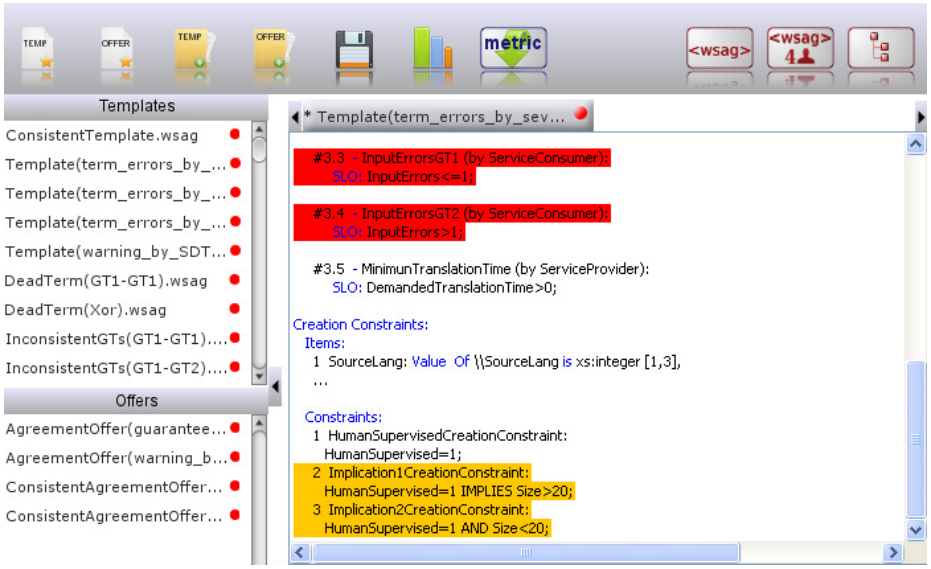

Fig. 5. Screenshot with the colored highlighting of conflicts

lighted with a different color for each set of terms that are in a conflict between themselves.

5.2 Getting Compliant WS-Agreement documents

Following with the same translation service scenario, once published a conflict-free template, the agreement offer of Document 1.3 may be sent to the provider from a third party interested in the service. At this moment, it is very appealing for the provider to check if the received offer is conflict-free and compliant with the template or not. This task is needed before achieving a final contract between parties to avoid problems during the service provisioning.

```
AgreementOffer 2 – TranslateDemand
  1 Initiator: IneedTranslation Corp.,
  2 ServiceProvider: AgreementResponder,
  3 TemplateId: 1,
  4 TemplateName: Translate it!
```

```
Service Description Terms:
  1 Translation_by_Computer – TranslationService1:
      Size = 20 – measured By MediumInteger,
      HumanSupervised = 1 – measured By BinaryInteger,
      DemandedTranslationTime = 2,
      Cost = 1 – measured By MediumInteger;

Service properties:
 1 SP1 – TranslationService1
  1.1 DemandedTranslationTime – measured By Time;

Guarantees:
  1 TranslationTimeOffer (By ServiceProvider):
    SLO: DemandedTranslationTime<=100 AND DemandedTranslationTime>0;
```

1.3. Ag. Offer of translation service scenario

Analysing with the editor the compliance between the offer 1.3 and a conflict-free template, the provider would see in the log console of Figure 6 that both documents are conflict-free, but they are not compliant. Then, as Figure 7 denotes, both documents are opened in parallel and the conflicts are highlighted in the agreement offer to make easier for the users to find the conflicts.

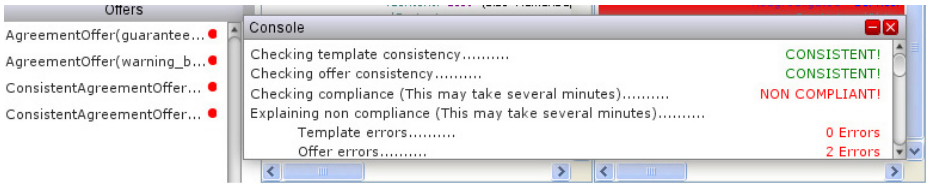

Fig. 6. log console screenshot to trace the non-compliance explaining

6 Conclusions and Future Work

In this paper, we argue that ADA is an useful framework for the WS–Agreement community because it complements other WS–Agreement frameworks such as WSAG4J by fulfilling the need for an analysis tool that automatically extracts useful information from agreement documents. Specifically, it provides (1) consistency and compliance checking and (2) the explanation of inconsistency and non-compliance conflicts of WS-Agreement documents specified either with the XML language of the recommendation [1] or WSAg4People, a plain-text language equivalent to the XML schema specified in [1]. Moreover, ADA has been carefully designed to make the development of new analysis operations easier.

Finally, ADA can be integrated with other tools and frameworks by means of three different interfaces: a Java (library), a Java (OSGi bundle), or a web service (WSDL). As example, we provide in this paper how the latter web service

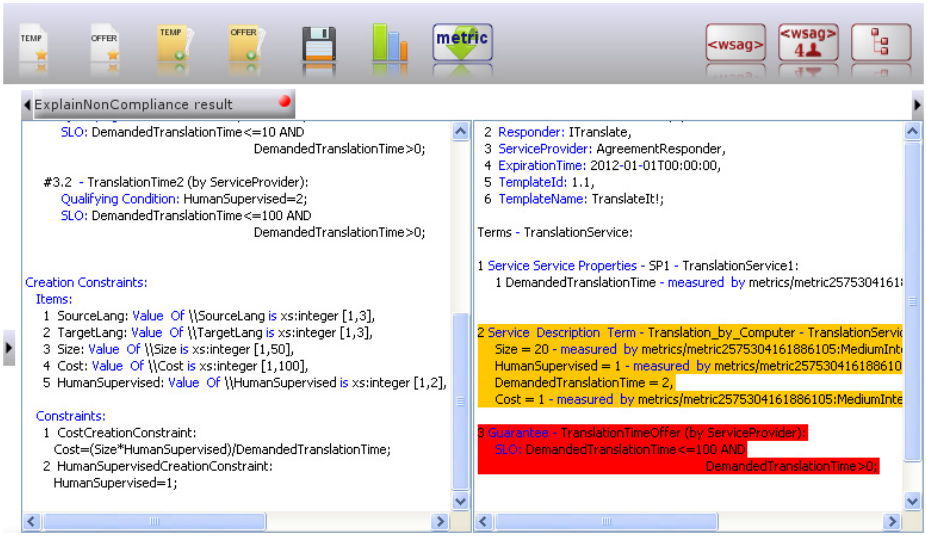

Fig. 7. Screenshot with the colored highlighting of conflicts in the agreement offer

interface is used from a developed rich internet application (ADA-FrontEnd) which is available for trying ADA at `http://www.isa.us.es/ada`. Moreover, we include some use cases of such editor to show how it helps to get conflict-free documents.

Our future work includes:

1. The integration of ADA with other WS-Agreement frameworks or tools such as: *WSAG4J* -ongoing work-, *SLA@SOI* [8] tools -ongoing work-, *Agent Scape*[11], *Akogrimo*[12], *Askalon GridARM*[13], *AssessGrid Negotiation Manager*[14], *BEinGRID Negotiation Manager*[15], *BREIN* [21], *CATNETS*[16], *Job Submission Service (JSS)*[17].

[11] Frances Brazier et al. from the Intelligent Interactive Distributed Systems group of the Delft University of Technology. `http://www.iids.org/research/aos`

[12] Access to Knowledge through the Grid in a mobile World (Akogrimo). `http://www.akogrimo.org/`

[13] Askalon. Thomas Fahringer et al. from the Distributed and Parallel Systems Group of the University of Innsbruck. `http://www.dps.uibk.ac.at/projects/askalon/`

[14] Agent Scape. `https://cit-server.cit.tu-berlin.de/trac/negmgr/wiki`

[15] BEinGRID SLA Negotiation component. Developed by the High Performance Computing Center Stuttgart (HLRS) in cooperation with Atos Origin Barcelona (ATOS). `https://gforge.beingrid.eu/gf/project/slanegotiator`

[16] Catallaxy paradigm for decentralized operation of dynamic application networks (CATNETS). `http://www.catnets.uni-bayreuth.de/`

[17] Job Submission Service (JSS). Umea University. `http://www.cs.umu.se/research/grid/jss/index.html`

2. The development of some new analysis operations such as the adaptation to WS-Agreement documents of the selection of the optimal agreement offer for a given template, started by us in [3].
3. The improvement of the temporal–awareness of WS–Agreement documents accepted by ADA, with the temporal extension we detailed in [15]. This would enable the analysis WS–Agreement documents with highly expressive periodical/non-periodical and disjoint/non-disjoint validity periods applied either to the whole document or concrete terms. For instance, on Monday-Friday from 8:00 to 18:00 the translation service availability decreases due to a higher number of requests; and the availability increases at home hours because of a lower number of requests (cf. [15], for details on XML validity periods definitions).

Acknowledgment

We would like to thank Pablo Trinidad Martín-Arroyo for his helpful comments about ADA structure; and Oliver Wäldrich and Wolfgang Ziegler for his helpful comments about ADA–WSAG4J synergy. We would also like to thank the technical staff for his work and dedication: Javier Martín de Agar Tirado, Antonio Jurado Serrano, Francisco López Sánchez, and specially to Jesús García Galán. Finally, we are grateful for the useful reviewers comments.

References

1. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS−Agreement) GDF−R−P.107. Open Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG, March 2007.
2. D. Battré, P. Wieder, and W. Ziegler. WS−Agreement Specification Version 1.0 Experience Document. Open Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG, August 2009.
3. A. Ruiz−Cortés, O. Martín−Díaz, A. Durán, and M. Toro. Improving the Automatic Procurement of Web Services using Constraint Programming. Int. Journal on Cooperative Information Systems, 14(4), 2005.
4. Nicole Oldham and Kunal Verma. Semantic WS-Agreement Partner Selection. In Proc. of the 15^{th} International World Wide Web Conference, pages 697−706. ACM Press, 2006.
5. G. Dobson and A. Sánchez−Macián. Towards Unified QoS/SLA Ontologies. In 3^{rd} IEEE Intl. ICWS/SCC Workshop on Semantic and Dynamic Web Processes, pages 169−174, Chicago, IL, September 2006. IEEE Computer Society Press.
6. S. Lamparter, S. Luckner, and S. Mutschler. Formal Specification of Web Service Contracts for Automated Contracting and Monitoring. In 40^{th} Hawaii International Conference on System Sciences. IEEE Computer Society Press, 2007.
7. M. Comuzzi and G. Spanoudakis. A framework for hierarchical and recursive monitoring of service based systems. In IEEE Intl. Conf. on Internet and Web Applications and Services, pages 383−388, 2009.

8. C. Kotsokalis, M.A. Rojas−González, K. Kearny, F. Torelli, M. Comuzzi, U. Winkler, T. Ellahi, A. Marconi, B. Fuentes, A. Castro−Escudero, L. Pasquale, and M. Evenson. SLA@SOI: SLA Management and Foundations. Technical Report Deliverable D.A5.a, SLA@SOI consortium, 2009.
9. Hua Xiao, B. Chan, Ying Zou, J.W. Benayon, B. O'Farrell, E. Litani, and J. Hawkins. A framework for verifying sla compliance in composed services. pages 457−464, Sept. 2008.
10. Congwu Chen, Lei Li, and Jun Wei. Aop based trustable sla compliance monitoring for web services. pages 225−230, Oct. 2007.
11. Omer F. Rana, Martijn Warnier, Thomas B. Quillinan, and Frances M. T. Brazier. Monitoring and reputation mechanisms for service level agreements. In Grid Economics and Business Models (GECON), pages 125−139, 2008.
12. O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, and D. Cojocarasu. Managing violations in service level agreements. pages 349−358, 2008.
13. G. Grabarnik, H. Ludwig, and L. Shwartz. Dynamic management of outsourced service processes x2019; qos in a service provider − service supplier environment. pages 81−88, april 2008.
14. C. Braga, F. Chalub, and A. Sztajnberg. A Formal Semantics for a Quality of Service Contract Language. Electronic Notes in Theoretical Computer Science, 203(7):103−120, April 2009.
15. C. Müller, O. Martín−Díaz, A. Ruiz−Cortés, M. Resinas, and P. Fernández. Improving Temporal−Awareness of WS−Agreement. In Proc. of the 5^{th} Int. Conf. on Service−Oriented Computing (ICSOC), pages 193−206. Springer Verlag, 2007.
16. C. Müller, A. Ruiz−Cortés, and M. Resinas. An Initial Approach to Explaining SLA Inconsistencies. In Proc. of the 6^{th} Int. Conf. on Service−Oriented Computing (ICSOC), volume 5364 of LNCS, pages 394−406, Sydney, Australia, Dec 2008. Springer Verlag.
17. C. Müller, M. Resinas, and A. Ruiz−Cortés. Explaining the Non−Compliance between Templates and Agreement Offers in WS−Agreement*. In Proc. of the 7^{th} Int. Conf. on Service−Oriented Computing (ICSOC), volume 5900 of LNCS, pages 237−252, Sweden, Stockholm, Nov 2009. Springer Verlag.
18. C. Müller, A. Durán, M. Resinas, A. Ruiz−Cortés, and O. Martín−Díaz. Experiences from building a WS–Agreement document analyzer tool (Including use cases in WS–Agreement and WSAg4People). Technical Report ISA−10−TR−03, ISA Research Group, July 2010, http://www.isa.us.es/modules/publications/getPdf.php?-idPublication=322.
19. E. Tsang, Foundations of Constraint Satisfaction, Academic Press, 1995.
20. O. Martín−Díaz, A. Ruiz−Cortés, A. Durán, and C. Müller. An approach to temporal aware procurement of web services. In Proc. of the 5^{th} Int. Conf. on Service−Oriented Computing (ICSOC), pages 170−184, 2005.
21. Business objective driven REliable and Intelligent grids for real busiNess (BREIN). 17 European partners. Final Brein Architecture document. 2009.