Explaining the Non-Compliance between Templates and Agreement Offers in WS-Agreement[*]

Carlos Müller, Manuel Resinas, Antonio Ruiz-Cortés

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

Abstract. A common approach to the process of reaching agreements is the publication of templates that guide parties to create agreement offers that are then sent for approval to the template publisher. In such scenario, a common issue the template publisher must address is to check whether the agreement offer received is compliant or not with the template. Furthermore, in the latter case, an automated explanation of the reasons of such non-compliance is very appealing. Unfortunately, although there are proposals that deal with checking the compliance, the problem of providing an automated explanation to the non-compliance has not yet been studied in this context. In this paper, we take a subset of the WS-Agreement recommendation as a starting point and we provide a rigorous definition of the explanation for the non-compliance between templates and agreement offers. Furthermore, we propose the use of constraint satisfaction problem (CSP) solvers to implement it and provide a proof-of-concept implementation. The advantage of using CSPs is that it allows expressive service level objectives inside SLAs.

Keywords: Service Level Agreement, SLA, WS-Agreement, Compliance Checking, Debugging, Quality of Service, Explanations.

1 Introduction

A common approach to the creation of agreements is by means of templates. For instance WS-Agreement specification [5] defines an XML-based language and a protocol for advertising the capabilities and preferences of services providers in templates, and creating agreements based on them. Specifically, WS-Agreement allows to specify templates that are published by a responder party, for instance an Internet service provider could have two public templates for a "basic" and

[*] This work has been partially supported by the European Commission (FEDER), Spanish Government under the CICYT projects Web-Factories (TIN2006-00472), and SETI (TIN2009-07366); and project P07-TIC-2533 funded by the Andalusian local Government.

a "premium" Internet service. A typical interaction process using templates and offers could be as follows: (1) an initiator party take a public template from a responder party, describing the agreement terms and some variability that must be taken into account by initiator in order to achieve an agreement; (2) an agreement offer may be sent to the responder party, including several changes, or not, into the initial template; (3) finally, the responder party may accept or not the agreement offer received. To use such approach of templates and offers, once established that the agreement offer is consistent [15], the problem is to ensure the compliance between agreement templates and offers. Some proposals such as [13,19] focus on checking whether an SLA is compliant with another one, and, hence, they could be adapted to check the compliance between agreement templates and offers. However, if they are not compliant, an explanation would make it easier to solve problems between parties. This explanation may be provided as the subset of terms of both template and agreement offer, that causes the non-compliance. For example, the Internet service provider could establish inside a template the bandwidth limit, allowing the user to customise of download and upload speeds as follows:

- Template: $\{t1 : downloadSpeed > 5Mb, t2 : uploadSpeed < 0.768Mb,$ $t3 : downloadSpeed + uploadSpeed < 5.768Mb\}$
- Agreement Offer: $\{o1 : downloadSpeed = 10Mb, o2 : uploadSpeed = 0.7Mb\}$

The explanation for the non-compliance of the previous example would be the following set of terms: $\{t3, o1, o2\}$.

Generally speaking, finding an explanation for the non-compliance is not as easy as in previous example. It is especially complex when a high expressiveness of the language used to specify the service terms is needed.

Solution overview and contribution: This paper is focused on providing explanations of the non-compliance between templates and agreement offers. To this end, we take our previous work in [15], in which we detail an approach to explain the inconsistencies in one SLA, as a starting point and we extend it to enable the checking of the compliance between templates and agreement offers and to provide explanations of the non-compliance.

Specifically, we extend the definition of the WS-Agreement subset of [15] to provide rigorous definitions of templates, the compliance between templates and offers and the explanation for the non-compliance. Then, we use such definitions to map agreement offers and templates into constraint satisfaction problems (CSPs) [21]. The CSP is sent to a constraint solver with an explanation engine [8,20] to get the terms that are causing the non-compliance. The advantage of using CSPs is that it allows the use of expressive assertions inside SLA terms, including arithmetic, comparison and logic operations such as $+, -, *, \div, >, \geq, <, \leq, \rightarrow, \ldots$. Furthermore, we have developed a proof-of-concept which is available for testing at `http://www.isa.us.es/wsag`.

The remainder of the paper is organized as follows: Section 2 describes the used subset of WS-Agreement in Section 2.1, rigorous definitions for agreement offers and templates in Section 2.2, the compliance between WS-Agreement* templates and offers in Section 2.3, and the explanation for the non-compliance between templates and offers in Section 2.4; Section 3 describes the process of explaining the non-compliance of WS-Agreement* templates and offers using CSP; Section 4 informs about the related work; and finally Section 5 conclude this paper anticipating some future work.

2 WS-Agreement*-Non-compliant Offers and Templates

2.1 WS-Agreement* Offers and Templates

Due to the flexibility and extensibility of WS-Agreement, we focus on WS-Agreement*, which is a subset of WS-Agreement (cf. `http://www.isa.us.es/wsag`, for details about these differences). WS-Agreement* just imposes several restrictions on some elements of WS-Agreement but it keeps the same syntax and semantics, therefore any WS-Agreement document that follows these restrictions is a WS-Agreement* document. Furthermore, note that, although WS-Agreement* is not as expressive as WS-Agreement, it does allow to express complex agreement documents as those in Figure 1, in which the elements of several WS-Agreement* documents in a computing services providing scenario are depicted. The complete XML documents are available at `http://www.isa.us.es/wsag`.

– **Name & Context** identifies the agreement and other information such as a template name and identifier, if any, referring to the specific name and version of the template from which the current agreement is created. For instance, context of Figure 1(c) refers to Template of Figure 1(a).
– **Terms** can be composed using the three term compositors described in [5]: `All` ($\wedge$), `ExactlyOne` ($\oplus$), and `OneOrMore` ($\vee$). All terms in the document must be included into a main `All` term compositor. Figure 1(a) includes `All` and `ExactlyOne` term compositors. Terms can be divided into:
  Service Terms including:
  • **Service properties** must define all variables that are used in the guarantee terms and other agreement elements, explained later. In Figure 1(a), the variables defined are the *availability of the computing service* (Availability), the *mean time between two consecutive requests of the service* (MTBR), and the *initial cost for the service* (InitCost). The type and general range of values for each variable is provided in an external document such as the ad-hoc XML document depicted in Figure 1(b).
  • **Service description terms** provide a functional description of a service, i.e. the information necessary to provide the service to the consumer. They may set values to variables defined in the service properties (e.g. InitCost=20 in Figure 1(a)) or they may set values to new variables. Type and domains are defined in external files such as XML Schemas (e.g. CPUsType=Cluster in Figure 1(a)).
  Guarantee terms describe the service level objectives (SLO) that a specific obligated party must fulfill, and a qualifying condition that specifies the validity condition under which the SLO is applied. For instance the Lower-Availability guarantee term included in Figure 1(a).

In [5], a WS-Agreement template is an agreement document with the structure of a WS-Agreement document described above, but including agreement creation constraints that should be taken into account during the agreement creation process. These `Creation Constraints` describe the variability allowed by the party who makes the template public. They include (1) general `Constraints` involving the values of one or more terms, for instance the FinalCost definition of "Constraint 1" of Figure 1(a); or (2) `Items` specifying that a particular variable of the agreement must be present in the agreement offer, typically as a service description term, and its range of values. For instance, the item elements of Figure 1(a) define three variables: the *number of Dedicated Central Processing Units* (CPUs), the *increase of the cost due to the selected MTBR* (ExtraMTBRCost), and the *final cost for the service* (FinalCost).

2.2 What's in WS-Agreement*?

To automate the explaining of the non-compliance, it is necessary to define the compliance between template and agreement offers and provide a rigorous definition of the explaining for the non-compliance. A first step toward this goal is to extend the definition of WS-Agreement* in [15] to provide rigorous definitions of templates, the compliance between templates and offers and the explanation for the non-compliance.

Definition 1 (A WS-Agreement* agreement offer). *A WS-Agreement* agreement offer α is a three-tuple composed of the variables defined in service properties and service description terms, their domains and a set of terms:*

$$\alpha = (\upsilon^\alpha, \delta^\alpha, T^\alpha), \text{ where}$$

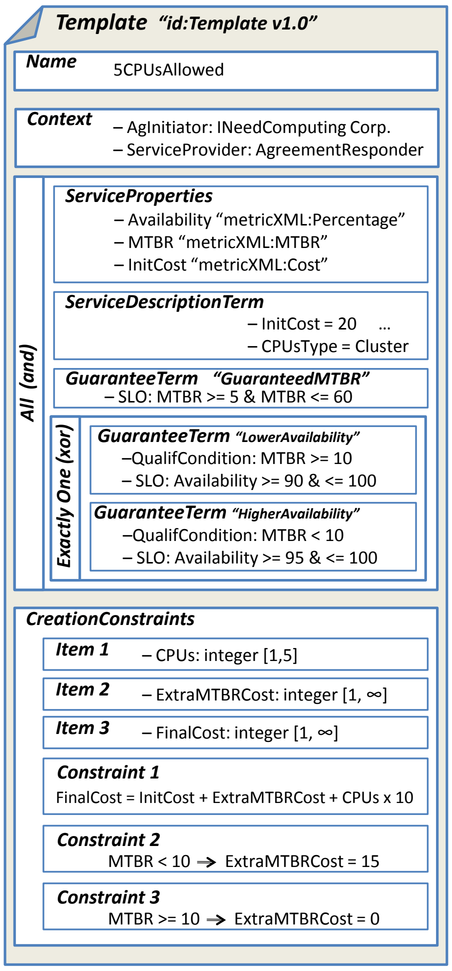
- $\upsilon^\alpha = \upsilon_p^\alpha \cup \upsilon_d^\alpha \neq \emptyset$ *is the finite set of variables defined in service properties* (υ_p^α)*, and in service description terms* (υ_d^α)*, respectively.*
- $\delta^\alpha = \delta_p^\alpha \cup \delta_d^\alpha \neq \emptyset$ *is the finite set of domains for those variables.*
- $T^\alpha = \{t_i^\alpha\}_{i=1}^n \neq \emptyset$ *is a finite set of terms, including service description terms, guarantee terms and terms compositors as follows:*

$$\textit{where } t_i^\alpha = \left\{ \begin{array}{lll} \lambda^\alpha = (\upsilon_i, value(\upsilon_i)) & \textit{if } t_i^\alpha \textit{ is a service description term} & (1) \\ \gamma^\alpha = (\kappa^\alpha(\upsilon), \sigma^\alpha(\upsilon)) & \textit{if } t_i^\alpha \textit{ is a guarantee term} & (2) \\ (t_{i1}^\alpha \wedge \ldots \wedge t_{im}^\alpha) & \textit{if } t_i^\alpha \textit{ is an } \texttt{All} \textit{ term compositor} & \\ (t_{i1}^\alpha \oplus \ldots \oplus t_{im}^\alpha) & \textit{if } t_i^\alpha \textit{ is an } \texttt{ExactlyOne} \textit{ term compositor} & \\ (t_{i1}^\alpha \vee \ldots \vee t_{im}^\alpha) & \textit{if } t_i^\alpha \textit{ is an } \texttt{OneOrMore} \textit{ term compositor} & \end{array} \right\}$$

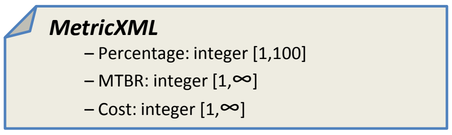
Where Clause (1) defines the value of variable ($value(\upsilon_i)$)*,* $\upsilon_i \in \upsilon^\alpha, value(\upsilon_i) \in \delta_i$*; and Clause (2) defines a guarantee term which includes:*

$$\kappa^\alpha(\upsilon) = \left\{ \begin{array}{l} true \textit{ if there is no qualifying condition or } (\forall \upsilon_i \in \upsilon^\alpha) \textit{ satisfies it} \\ false \textit{ otherwise} \end{array} \right\}$$

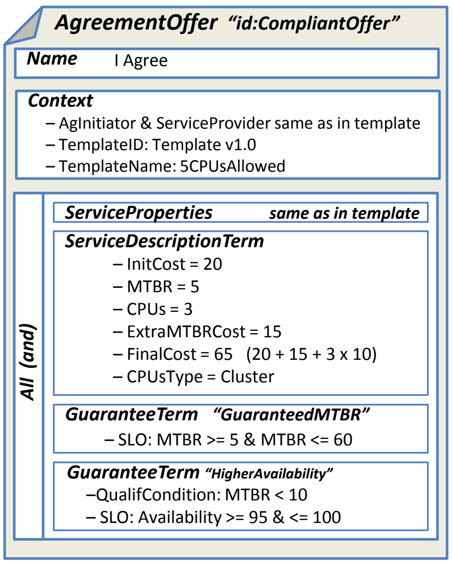
$$\sigma^\alpha(\upsilon) = \left\{ \begin{array}{l} true \textit{ if } (\forall \upsilon_i \in \upsilon^\alpha) \textit{ satisfies the SLO} \\ false \textit{ otherwise} \end{array} \right\}$$


Template "id:Template v1.0"
Name 5CPUsAllowed
Context – AgInitiator: INeedComputing Corp.
– ServiceProvider: AgreementResponder
All (and)
ServiceProperties
– Availability "metricXML:Percentage"
– MTBR "metricXML:MTBR"
– InitCost "metricXML:Cost"
ServiceDescriptionTerm
– InitCost = 20 ...
– CPUsType = Cluster
GuaranteeTerm "GuaranteedMTBR"
– SLO: MTBR >= 5 & MTBR <= 60
Exactly One (xor)
GuaranteeTerm "LowerAvailability"
–QualifCondition: MTBR >= 10
– SLO: Availability >= 90 & <= 100
GuaranteeTerm "HigherAvailability"
–QualifCondition: MTBR < 10
– SLO: Availability >= 95 & <= 100
CreationConstraints
Item 1 – CPUs: integer [1,5]
Item 2 – ExtraMTBRCost: integer [1, ∞]
Item 3 – FinalCost: integer [1, ∞]
Constraint 1
FinalCost = InitCost + ExtraMTBRCost + CPUs x 10
Constraint 2
MTBR < 10 → ExtraMTBRCost = 15
Constraint 3
MTBR >= 10 → ExtraMTBRCost = 0

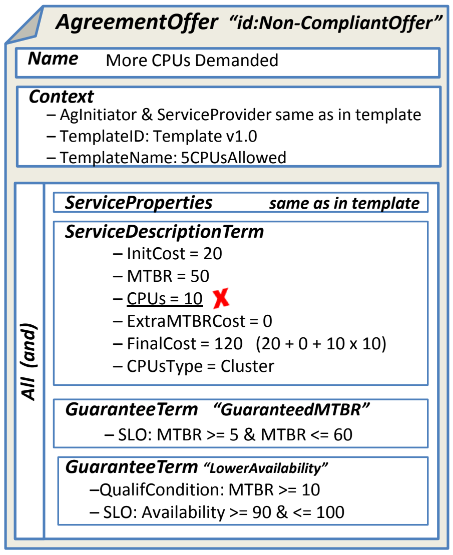
(a) A WS-Agreement template with general and item constraints.



(b) Content of the ad-hoc XML document for the variable domains of Figures "a", "c", and "d".


AgreementOffer "id:CompliantOffer"
Name I Agree
Context
– AgInitiator & ServiceProvider same as in template
– TemplateID: Template v1.0
– TemplateName: 5CPUsAllowed
All (and)
ServiceProperties same as in template
ServiceDescriptionTerm
– InitCost = 20
– MTBR = 5
– CPUs = 3
– ExtraMTBRCost = 15
– FinalCost = 65 (20 + 15 + 3 x 10)
– CPUsType = Cluster
GuaranteeTerm "GuaranteedMTBR"
– SLO: MTBR >= 5 & MTBR <= 60
GuaranteeTerm "HigherAvailability"
–QualifCondition: MTBR < 10
– SLO: Availability >= 95 & <= 100

(c) A Compliant offer with template "a".


AgreementOffer "id:Non-CompliantOffer"
Name More CPUs Demanded
Context
– AgInitiator & ServiceProvider same as in template
– TemplateID: Template v1.0
– TemplateName: 5CPUsAllowed
All (and)
ServiceProperties same as in template
ServiceDescriptionTerm
– InitCost = 20
– MTBR = 50
– CPUs = 10 ✗
– ExtraMTBRCost = 0
– FinalCost = 120 (20 + 0 + 10 x 10)
– CPUsType = Cluster
GuaranteeTerm "GuaranteedMTBR"
– SLO: MTBR >= 5 & MTBR <= 60
GuaranteeTerm "LowerAvailability"
–QualifCondition: MTBR >= 10
– SLO: Availability >= 90 & <= 100

(d) A Non-Compliant offer with template "a", demanding more dedicated CPUs.

Fig. 1. Template and Offers WS-Agreement* documents.

For the scenario of Figure 1(c), $\upsilon_p^\alpha = \{$ `Availability`, `MTBR`, `InitCost` $\}$, with theirs domains δ_p^α defined in Figure 1(b); $\upsilon_d^\alpha = \{$ `CPUsType` $\}$ with a domain δ_d^α defined in an XML-Schema (cf. Section 2.1); and $T^\alpha = \{$ λ_1^α:`InitCost=20` $\wedge$ λ_2^α :`MTBR=5` $\wedge$ λ_3^α :`CPUs=3` $\wedge$ λ_4^α :`ExtraMTBRCost=15` $\wedge$ λ_5^α :`FinalCost=65` $\wedge$ λ_6^α :`CPUsType=Cluster` $\wedge$ γ_1^α :$(\kappa_1^\alpha = \emptyset) \Rightarrow (\sigma_1^\alpha = MTBR >= 5$ `&` $MTBR <= 60) \wedge \gamma_2^\alpha$:$(\kappa_2^\alpha = MTBR < 10) \Rightarrow (\sigma_2^\alpha = Availability >= 95$ `&` $Availability <= 100)$ $\}$.

Following definition 1, we can define a WS-Agreement* template, excluding name and context elements, as follows:

Definition 2 (A WS-Agreement* template). *A WS-Agreement* template* θ *is a four-tuple of the form:*

$$\theta = \left(\upsilon^\theta, \delta^\theta, T^\theta, \phi^\theta(\upsilon^\theta)\right), \ \textit{where}$$

- $\upsilon^\theta = \upsilon_p^\theta \cup \upsilon_d^\theta \cup \upsilon_c^\theta \neq \emptyset$ *is the finite set of variables defined in service properties* (υ_p^α)*, and in service description terms* (υ_d^α)*, and in items of creation constraints* (υ_c^θ)*, respectively.*
- $\delta^\theta = \delta_p^\theta \cup \delta_d^\theta \cup \delta_c^\theta \neq \emptyset$ *is the finite set of domains for those variables.*
- $T^\theta = \{t_i^\theta\}_{i=1}^n \neq \emptyset$ *is a finite set of terms* $\equiv T^\alpha$ *but applied to templates instead of agreement offers.*
- $\phi^\theta : (\delta_1^\theta \times \ldots \times \delta_n^\theta) \rightarrow \{true, false\}$ *is a function defined as follows:*

$$\phi^\theta(\upsilon_1, \ldots, \upsilon_n) = \begin{Bmatrix} \textit{true if } (\upsilon_1, \ldots, \upsilon_n) \textit{ satisfies all constraints} \\ \textit{false otherwise} \end{Bmatrix}$$

For the scenario of Figure 1(a), $\upsilon_p^\theta = \{$ `Availability`, `MTBR`, `InitCost` $\}$, with theirs domains δ_p^θ defined in Figure 1(b); $\upsilon_d^\theta = \{$ `CPUsType` $\}$ with a domain δ_d^θ defined in an XML-Schema; $\upsilon_c^\theta = \{$ `CPUs`, `ExtraMTBRCost`, `FinalCost` $\}$ with its domain δ_c^θ defined in each item; $T^\theta = \{$ λ_1^θ:`InitCost=20` $\wedge$ λ_2^θ :`CPUsType = Cluster` $\wedge$ γ_1^θ :$(\kappa_1^\theta = \emptyset) \Rightarrow (\sigma_1^\theta = MTBR >= 5$ `&` $MTBR <= 60) \wedge (\gamma_2^\theta$:$(\kappa_2^\theta = MTBR >= 10) \Rightarrow (\sigma_2^\theta = Availability >= 90$ `&` $Availability <= 100) \oplus \gamma_3^\theta$:$(\kappa_3^\theta = MTBR < 10) \Rightarrow (\sigma_3^\theta = Availability >= 95$ `&` $Availability <= 100)$ $)\}$; and $\phi^\theta(\upsilon^\theta) = Constraint1 \wedge Constraint2 \wedge Constraint3 = (FinalCost = InitCost{+}ExtraMTBRCost{+}CPUs{\times}10) \wedge (MTBR < 10 \Rightarrow \text{ExtraMTBRCost} = 15) \wedge (MTBR >= 10 \Rightarrow ExtraMTBRCost = 0)$.

2.3 Compliance between Templates and Agreement Offers

In WS-Agreement [5] the compliance of offers with templates is defined as follows:

*"**Agreement template compliance**: An agreement offer is compliant with a template advertised by an agreement responder if and only if each term of service described in the `Terms` section of the agreement offer complies with the term constraints expressed in the `CreationConstraints` section of the agreement template. In addition, in the `Context` of the offer, the `Agreement Responder` value must match the value specified in the template; and the `Template Id` must exactly match the name provided in the template document against which compliance is being checked."*

This compliance is summarised with discontinuous arrows in Figure 2. Note that this definition of compliance does not state anything about the terms of the template. In other words, the party that creates the agreement offer may ignore the terms specified in the template. The problem with this definition is that the template creator can specify terms in the template, but the party that creates the agreement offer cannot do anything with them because the definition of compliance does not provide any semantics with regard to them. Thus, it is unknown for the party that creates the agreement offer whether the terms of the template specify default values, or preferred values, or mandatory values that could not be expressed by means of creation constraints, or any other meaning.

To solve this issue, we provide an extended definition of compliance, the so-called t-compliance, that extends the previous definition of compliance with the requirement that the terms of the agreement offer must be compliant with the terms of the template. This is depicted in Figure 2 by means of continuous arrows.

This new notion of compliance raises another issue: *does the compliance between the terms of the agreement offer and the terms of the template implies that agreement offer terms must syntactically match with template terms or they must match semantically?*

A syntactic match means that terms that appear in the template must appear as is in the agreement offer, perhaps after selecting some of the alternatives provided by the term compositors. For instance, the guarantee terms of the agreement offer of Figure 1(c) syntactically matches the guarantee terms of the template of Figure 1(a).

A semantic match means that all possible assignment of values to the variables that satisfies the terms of the template must satisfy the terms of the agreement offer. as well. For instance, the guarantee term $MTBR >= 3$ `&` $MTBR <= 60$ semantically matches the guarantee term `GuaranteedMTBR` of the template. In this paper we choose the semantic match because syntactic match is just a particular case of semantic match.

Then, assuming the context compliance between documents, we can define the compliance and t-compliance between WS-Agreement* offers and templates. But previously we define an auxiliary operation to represent if a vector of value assignments to all variables `satisfies` a concrete term.

Definition 3 (Satisfies Operation: $satisfies(t_i, \upsilon)$**).**
We define operation $satisfies(t_i, \upsilon)$*, as a function such that, given a term* t_i *and a vector* $(\upsilon_1, \ldots, \upsilon_n)$ *of value assignments to all variables, it returns true if* $(\upsilon_1, \ldots, \upsilon_n)$ *satisfies the term and false, otherwise:*

$$satisfies : T \times (\delta_1 \times \ldots \times \delta_n) \rightarrow \{true, false\}, \text{ where}$$

$$satisfies(t_i, \upsilon) \Leftrightarrow \left\{ \begin{array}{ll} \upsilon_i = value(\upsilon_i) & (1) \\ \sigma(\upsilon) & (2) \\ \kappa(\upsilon) \Rightarrow \sigma(\upsilon) & (3) \\ \bigwedge_{i=1}^{n} satisfies(t_i, \upsilon) & (4) \\ \bigwedge_{i=1}^{n} satisfies(t_i, \upsilon) \Leftrightarrow (\bigwedge_{j=1 \backslash j \neq i}^{k} \neg satisfies(t_i, \upsilon)) & (5) \\ \bigvee_{i=1}^{n} satisfies(t_i, \upsilon) & (6) \end{array} \right\}$$

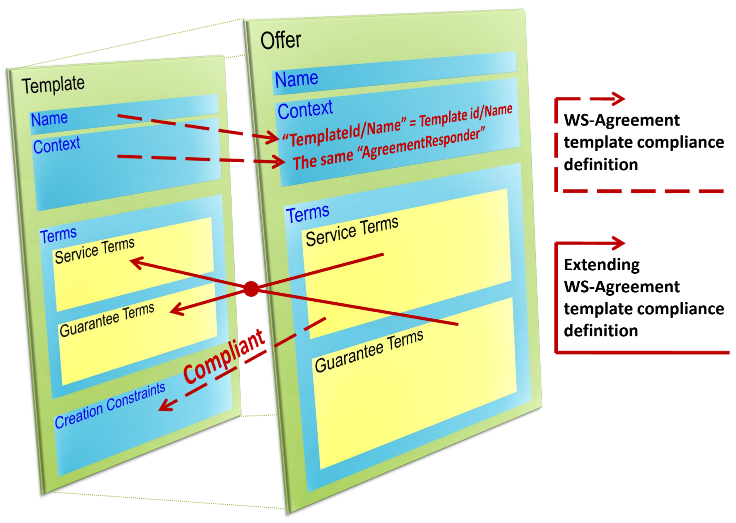

Template
Name
Context
Terms
Service Terms
Guarantee Terms
Creation Constraints
Compliant
Offer
Name
Context
"TemplateId/Name" = Template id/Name
The same "AgreementResponder"
Terms
Service Terms
Guarantee Terms
WS-Agreement template compliance definition
Extending WS-Agreement template compliance definition

Fig. 2. Summary of Compliance between WS-Agreement templates and offers.

Clause (1) is applied when t_i *is a service description term* $\lambda = (\upsilon_i, value(\upsilon_i))$. *Clause (2) is applied if* t_i *is a guarantee term without qualifying condition* $\gamma = (\emptyset, \sigma)$. *Clause (3) is applied if* t_i *is a guarantee term with qualifying condition* $\gamma = (\kappa, \sigma)$. *And Clauses (4, 5, and 6) are applied if* t_i *is an* `All`$(\wedge)$, `ExactlyOne`$(\oplus)$, *and* `OneOrMore`$(\vee)$ *term compositor, respectively.*

Definition 4 (WS-Agreement* template compliance).
A WS-Agreement offer* $\alpha = (\upsilon^\alpha, \delta^\alpha, T^\alpha)$ *is compliant with a WS-Agreement* template* $\theta = (\upsilon^\theta, \delta^\theta, T^\theta, \phi^\theta(\upsilon^\theta))$, *iff the following operation is true:*

$$compliance(\alpha, \theta) \Leftrightarrow \left\{ \begin{array}{ll} \quad \upsilon_p^\theta = \upsilon_p^\alpha \wedge \delta_p^\theta = \delta_p^\alpha \wedge & (1) \\ \wedge\ \upsilon_d^\alpha = \upsilon_c^\theta \cup \upsilon_d^\theta \wedge \delta_d^\alpha = \delta_c^\theta \cup \delta_d^\theta \wedge & (2) \\ \wedge\ matches(T^\alpha, \phi^\theta) & (3) \end{array} \right\}$$

where $matches(T^\alpha, \phi^\theta) \Leftrightarrow \forall v \in (\delta_1 \times \ldots \times \delta_n)$, $\phi^\theta(v) = true \Rightarrow (\ \forall t_i \in T^\alpha$, $matches(t_i, v)\ )$.

Clause (1) means that variables and domains defined inside service properties of a compliant agreement offer must be the same as defined inside template. Clause (2) ensures that all variables and domains defined inside service description term of a compliant agreement offer are defined inside service description term of template or inside item element of template creation constraints. This does not allow to add any more variables and domains inside service description terms of a compliant agreement offer to such defined in template. Finally, Clause (3) means that each terms of a compliant agreement offer must match general constraints of template creation constraints.

Definition 5 (WS-Agreement* template t-compliance).
A WS-Agreement offer $\alpha = (\upsilon^\alpha, \delta^\alpha, T^\alpha)$ is t-compliant with a WS-Agreement* template $\theta = \big(\upsilon^\theta, \delta^\theta, T^\theta, \phi^\theta(\upsilon^\theta)\big)$, iff the following operation is true:*

$$\textit{t-compliance}(\alpha, \theta) \Leftrightarrow \textit{compliance}(\alpha, \theta) \textit{ AND matches}(T^\alpha, T^\theta)$$

where matches(T^α, T^θ) $\Leftrightarrow \forall\ v \in (\delta_1 \times \ldots \times \delta_n)$, ($\forall t_j \in T^\theta$, matches(t_j, υ)) $\Rightarrow$ ($\forall t_i \in T^\alpha$, matches(t_i, υ)). In other words, each term of a compliant agreement offer must match template terms.

Figure 1(c) and 1(d) depict two possible responses for the agreement template of Figure 1(a). Figure 1(c) is a compliant agreement offer because all template general constraints are taken into account for the agreement offer service description term specification (clause (3) of compliance definition); and it is a t-compliant offer because it does not include neither different value definitions for variables, nor any term which were not semantically matched with template terms (t-compliance definition). However, Figure 1(d) depicts a non-compliant agreement offer, and the explanation for such non-compliance must be provided. Note that we do not detail yet the explanation for the non-compliance to highlight the advantages of having a system capable of providing them.

2.4 Explaining the Non-Compliance

We consider an explanation for a non-compliance between agreement offers and templates as a minimum set of terms of both agreement offer and template that makes them not compliant. However, before defining rigorously the explanation, we must define two auxiliary operations.

Definition 6 (Closure of a set of terms: T^*).
The closure of a terms set (T^*) is the set of all possible agreements that can be obtained after selecting all the alternatives provided by the term compositors (All, ExactlyOne, and OneOrMore). T^* can be obtained by appliying the closure to non-composite terms (t_i^*), `All` term compositor (AND^*), `ExactlyOne` term compositor (XOR^*), and `OneOrMore` term compositor (OR^*) as follows:

$$T^* \Leftrightarrow \left\{ \begin{array}{l} t_i^* = \{\{t_i\}\} \\ AND^*(t_1, \ldots, t_n) = \{\{i_1 \cup \ldots \cup i_n\} | i_1 \in t_1^* \wedge \ldots \wedge i_n \in t_n^*\} \\ XOR^*(t_1, \ldots, t_n) = \bigcup_{i=1}^n t_i^* \\ OR^*(t_1, \ldots, t_n) = \bigcup_{p \in P(\{t_1, \ldots, t_n\}) - \emptyset} \{\{i_1 \cup \ldots \cup i_n\} | \\ \qquad\qquad\qquad\qquad | i_1 \in p_1^* \wedge i_n \in p_n^* \wedge p = \{p_1, \ldots p_n\}\} \end{array} \right)$$

Where P(S) is the power set of S.

For example, the closure of template of Figure 1(a) is: $T^{\theta *} = \{\{$ Init-Cost=20, CPUsType=Cluster, GuaranteedMTBR, LowerAvailability $\}\{$ Init-Cost=20, CPUsType=Cluster, GuaranteedMTBR, HigherAvailability $\}\}$.

Definition 7 (Terms Extraction Operation: $terms(T)$**).**
We define operation terms(T), where T is a set of terms including service description terms, guarantee terms, and term compositors; as an operation which obtain the set of service descriptions and guarantee terms of T.

This operation applied to template of Figure 1(a) is: $terms(T^{\theta}) = \{$ InitCost=20, CPUsType=Cluster, GuaranteedMTBR, LowerAvailability, HigherAvailability$\}$.

Finally, the explanation could be rigorously defined, using the `closure` definition and `terms(T)` operation, as follows:

Definition 8 (Explanation for WS-Agreement* template non-compliance).
Given a WS-Agreement offer* $\alpha = (\upsilon^{\alpha}, \delta^{\alpha}, T^{\alpha})$ *which is non-compliant with a WS-Agreement* template* $\theta = \left(\upsilon^{\theta}, \delta^{\theta}, T^{\theta}, \phi^{\theta}(\upsilon^{\theta})\right)$ *(i.e.* $\neg compliance(\alpha, \theta)$*), the explanation (E) is a minimal subset of terms defined as follows:*

$E = \epsilon^{\alpha} \cup \epsilon^{\theta} \cup \epsilon^{\phi}$*, where* $\epsilon^{\alpha} \in P(terms(T^{\alpha}) - \emptyset)$*,* $\epsilon^{\alpha} \subseteq n \in T^{\alpha *}$*, and* $\epsilon^{\theta} \in P(terms(T^{\theta}) - \emptyset)$*,* $\epsilon^{\theta} \subseteq n \in T^{\theta *}$*, and* $\epsilon^{\phi} \in P(\phi^{\theta})$*. Where P(S) is the power set of S.*

In other words, E is a minimal subset of conflictive terms extracted from the agreement offer terms, template terms and template creation constraints.

In the non-compliance between Figures 1(a) and 1(d), the resulting explanation would be: $\epsilon^{\phi} = \{$Item 1$\}$, and $\epsilon^{\alpha} = \{$CPUs=10$\}$. In such term the consumer is demanding more dedicated CPUs than the allowed by the provider template. Such underlined terms and the domain defined inside "Item 1" are the origin for the non-compliance situation and they are considered as the `explanation` for the non-compliance between such offer and template.

Other examples of non-compliance in the example of Figure 1(a) and 1(d), would be the following: (a) if we change the value of `CPUsType` inside the agreement offer there will be two different values for the same variable; (b) if we change the value of `ExtraMTBRCost` inside service description term of the agreement offer, it there will be in conflict with the `Constraint 3` of template; if we change the guarantee term `MTBRDomain` in the agreement offer, there will be in conflict with such guarantee term definition inside template.

The complexity of automating the search for explanations depends on the expressiveness of the language used to specify the agreement terms. An approach to automate this search is by means of constraint satisfaction problems (CSPs) and it is detailed in the following section.

3 Explaining The Non-Compliance using CSPs

3.1 Preliminaries

Constraint Satisfaction Problems Constraint Satisfaction Problems (CSP) [21] have been an object of research in Artificial Intelligence over the last few decades. A CSP is a three–tuple of the form (V, D, C) where $V \neq \emptyset$ is a finite set of variables, $D \neq \emptyset$ is a finite set of domains (one for each variable) and C is a constraint defined on V. Consider, for instance, the CSP:

$(\{a, b\}, \{[0, 2], [0, 2]\}, \{a + b < 4\})$. The solution of such CSP is whatever valid assignment of all elements in V that satisfies C. $(2, 0)$ is a possible solution of previous example since it verifies that $2 + 0 < 4$.

3.2 Mapping WS-Agreement* templates onto CSP

In [15] we define the mapping (μ) of a WS-Agreement* offer document (α) onto an equivalent CSP, (ψ^{α}). The variables (υ) defined inside the service properties are the CSP variables; the variable domains (δ) included in the document specified by the metric attribute are the CSP variable domains; and the constraints from the service description terms (λ_{υ}), guarantee terms (γ) and term compositors ($\wedge$ as a logic "AND", $\oplus$ as logic "XOR", and $\vee$ as logic "OR") are the CSP constraints.

Then, we have to study now how the creation constraints mapping should be included in order to get a complete WS-Agreement* template to CSP mapping. Figure 3 summarizes how the creation constraints, expressed as items are mapped as CSP variables (υ_c) and domains (δ_c); and expressed as general constraints (ϕ) are mapped as CSP constraints.

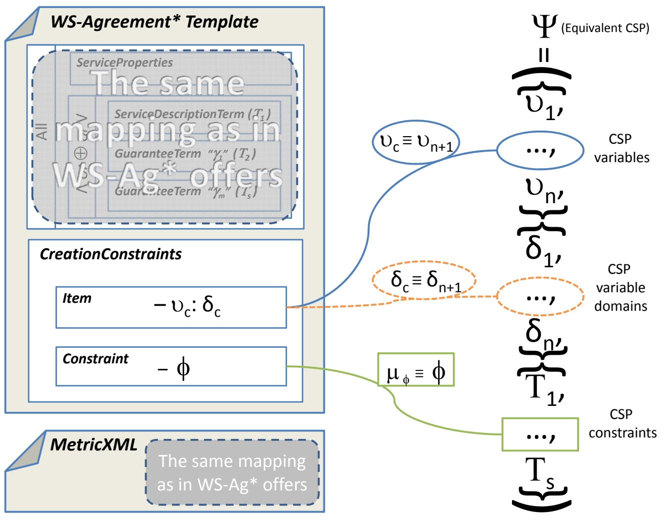


Fig. 3. Summary of WS-Agreement* template to CSP mapping.

Thus, in general, our WS-Agreement* template to CSP mapping can be defined as follows:

Definition 9 (Mapping an WS-Agreement* template to CSP). *The mapping* $(\mu : \theta \rightarrow \psi)$ *of a WS-Agreement* template* (θ) *to a CSP* (ψ) *can* *be defined as follows:*

$$\mu(\theta) = \mu\left(\upsilon_i, \delta_i, T_i, \phi_i\right) = \left(\{\upsilon_i\}, \{\delta_i\}, \{\mu_T(T_i)\}, \{\mu_\phi(\phi_i)\}\right) = \psi^\theta$$

where $\mu_\phi : \phi \rightarrow C$ is a direct mapping function of WS-Agreement general constraints into constraints, defined as follows: $\mu_\phi \equiv \{\phi\}$, and where $\mu_T : T \rightarrow C$ is a mapping function of terms into constraints defined in [15].*

Using the previous mapping, the ψ^θ for the template of Figure 1(a) is mapped as follows: (1) a set of variables where the three last are mapped from the creation constraints *{ Availability, MTBR, InitCost, CPUsType, CPUs, ExtraMTBRCost, FinalCost }*; (2) a set of domains for such variables *{ [1 ... 100], [1 ... ∞), [1 ... ∞), [Cluster, Multicore, Distributed], [1 ... 5], [1 ... ∞), [1 ... ∞) }*; and (3) a set of constraint where the three last are mapped from the creation constraints *{ InitCost = 20, CPUsType = Cluster, MTBR $\geq$ 5 $\wedge$ MTBR $\leq$ 60, ((MTBR $\geq$ 10) $\Rightarrow$ (Availability $\geq$ 90 $\wedge$ Availability $\leq$ 100)) $\Leftrightarrow$ $\neg$ ((MTBR < 10) $\Rightarrow$ (Availability $\geq$ 95 $\wedge$ Availability $\leq$ 100)) $\wedge$ ((MTBR < 10) $\Rightarrow$ (Availability $\geq$ 95 $\wedge$ Availability $\leq$ 100)) $\Leftrightarrow$ $\neg$ ((MTBR $\geq$ 10) $\Rightarrow$ (Availability $\geq$ 90 $\wedge$ Availability $\leq$ 100)), FinalCost = InitCost + ExtraMTBRCost + CPUs $\times$ 10, (MTBR < 10) $\Rightarrow$ (ExtraMTBRCost = 15), (MTBR $\geq$ 10) $\Rightarrow$ (ExtraMTBRCost = 0) }*

3.3 Explaining The Non-Compliance between WS-Agreements* Documents

To perform the explaining of the Non-Compliance between templates and agreement offers, we have developed aa proof-of-concept implementation which is available at `http:\\www.isa.us.es\wsag`. The input to the system is threefold: the WS-Agreement* offer, the WS-Agreement* template, and the XML document with the metrics of service properties. The whole process implemented by the proof-of-concept involves four parts:

1. A simple checking of the document contexts is carried out to ensure that the offer refers to the template that has been provided. If an error is returned, it must be reported to user.
2. Each WS-Agreement* documents are mapped into a CSP: (1) the CSP mapped from the WS-Agreement* offer $(V^\alpha, D^\alpha, C^\alpha)$, as defined in [15]; and (2) the CSP mapped from the WS-Agreement* template $(V^\theta, D^\theta, C^\theta)$, as defined in Section 3.2. To explain the non-compliance between both CSPs we have to join them in an unique CSP as it is described in [19]: $(V^\alpha \cup V^\theta, D^\alpha \cup D^\theta, C^\theta \rightarrow C^\alpha)$. Once the joined CSP is generated, we can check if it can be solved or not using CSP solvers. In the former case both documents are compliant.
3. An explanation engine obtains the explanations for the unsolved CSP and they are sent to the last part of our process.
4. Finally, a tracing component converts the explanations into the equivalent original agreement terms in order to classify the error to be reported to the user. The possible types of errors returned are:

– If the explanations involve terms from both documents, then there is a non-compliance between them.
– If the explanations involve terms from only one document, then this document is inconsistent.

For instance, if we check the disagreement between the non-compliant agreement offer of Figure 1(d) and the template of Figure 1(a), the first part would be passed due to the correct offer context. However, the explainer part will return, a minimal subset of the conflicting elements. Such elements are the underlined service description term of the offer against the item element of template creation constraint which detail the possible values for the dedicated CPUs. Then, the minimal subset of the example would be "CPUs = 10" and "CPUs >= 1 and CPUs <= 5". Each previous constraint would be traced back to its respective agreement element. In this case the constraints are traced back to the CPUs service description term inside offer and the CPUs item element inside template. Since the two conflictive elements come from the two agreement documents, the type of error occurred is a non-compliance between them.

4 Related Work

As far as we know, there are no proposals that deal with providing explanations for the non-compliance between agreement documents. This paper extends with template elements the definition of the WS-Agreement subset of [15] in which a first approach to explaining SLA inconsistencies was proposed. Previously, in [19], we studied mapping SLAs to CSPs, aimed at checking their consistency and conformance, which is a synonym of compliance. However, in that paper no explanation about the inconsistency or non-conformance of the documents was provided. In addition, [19] dealt with its own SLA specification instead of using a proposed standard format such as WS-Agreement.

Some proposals with similarities with our paper in their problem domain are the following ones: (1) The closest problem tackled in a research work is [16], in which Oldham et al. create a description logic-based ontology of WS-Agreement that could be used to check consistency and conformance of SLAs using a description logic reasoner. However, the authors do not detail what the consistency or conformance checking process is. Furthermore, they do not support the explanations for the inconsistent or non-conform terms. (2) A second group of proposals with some similarities in their problem domain deal with web service monitoring. For instance [22] checks the SLA compliance of web services compositions at a design time, but only for concrete types of SLOs and without providing any explanation for the non-compliance; [7] proposes a framework to audit if the execution of a web service is compliant with an unique SLA; [4] proposes the use of aspect oriented programming to monitor a concrete type of variables of an SLA; and [12] proposes a solution for managing SLAs in composite services. However, neither of them provide any explanation for the non-compliance. (3) Finally, [18] deals with the problem of compliance between SLOs and penalty clauses of an SLA, classifying the possible situations and using

WS-Agreement as case study, but again without providing any explanations for the non-compliance.

Other proposals with similarities with our paper but in their solution domains are the following: (1) The closest solution used in a research work is [1], in which Aiello et al. uses rigorous definitions about WS-Agreement element such as terms, agreement, and several states because they study the different agreement states of an agreement process. (2) There are many authors that deal with constraint-based paradigms to tackle different SLA aspects as for instance: in [9,10] constraint-based problem are used to solve web services requests in a web services interaction process; in [3] a constraint-based language is proposed to specify SLAs; in [2] constraints are used to optimize web services composition taking into account quality of service. However the scope of these works is completely different in comparison with this paper because they do not provide any explanation for the non-compliance between agreement documents. (3) A third group of proposals deal with explanation-based solution for the following problems: [17] proposes an explanation-based tool to be integrated into solvers and make the detection of conflicts more user-friendly, and [6,11] improves the use of explanations to perform the solution of CSPs more efficient.

5 Conclusions and Future Work

In this paper we have motivated the need for explaining the non-compliance between WS-Agreement documents and we have presented a first approach to reach this goal in an automated manner. More specifically, we present the problem of explaining the non-compliance in an implementation-independent manner using rigorous definitions for agreement offers, templates, their compliance, and the explanation for their non-compliance. Then we propose to map templates and agreement offers into a constraint satisfaction problem (CSP), in order to use a CSP solver together with an explanation engine to perform the compliance checking and return the non-compliant terms in an automated manner.

In summary, this paper provides the following contributions:

1. A rigorous definition of compliance between WS-Agreement* templates and offers. Additionally, the rigorous definition of compliance has allowed us to extend template compliance definition of WS-Agreement.
2. A rigorous definition of explanations for the non-compliance between WS-Agreement* templates and offers.
3. A description of a process that materialises the previous definitions by means of a constraint satisfaction problem (CSP) solver combined with an explanation engine.

Finally, we have developed a proof-of-concept implementation that is available at `http://www.isa.us.es/wsag`.

However, there are still some open issues that require further research: first, extending the rigorous definitions and the mapping to CSPs to full WS-Agreement specification; second, checking the consistency and compliance of WS-Agreement documents with the temporal extension we detailed in [14].

References

1. M. Aiello, G. Frankova, and D. Malfatti. What's in an Agreement? An Analysis and an Extension of WS-Agreement. In *ICSOC*, pages 424–436. Springer, 2005.
2. M. Alrifai and T. Risse. Combining global optimization with local selection for efficient qos-aware service composition. In *18th WWW Conf.*, pages 881–881, 2009.
3. M. G. Buscemi and U. Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In *ESOP, 4421 of LNCS*, pages 18–32, 2007.
4. Congwu Chen, Lei Li, and Jun Wei. Aop based trustable sla compliance monitoring for web services. pages 225–230, Oct. 2007.
5. Andrieux et al. of the OGF Grid Resource Allocation Agreement Protocol WG. Web Services Agreement Specification (WS-Agreement) (v. gfd.107), 2007.
6. D. Grimes. Automated within-problem learning for constraint satisfaction problems. 2008.
7. Hasan and Burkhard Stiller. Auric: A scalable and highly reusable sla compliance auditing framework. pages 203–215, 2007.
8. N. Jussien and V. Barichard. The PaLM system: explanation-based constraint programming. In *Proceedings of TRICS, pages = 118–133, year = 2000.*
9. A. Lazovik, M. Aiello, and R. Gennari. Encoding requests to web service compositions as constraints. In *In Constraint Programming*, pages 05–40, 2005.
10. A. Lazovik, M. Aiello, and R. Gennari. Choreographies: using constraints to satisfy service requests. pages 150–150, Feb. 2006.
11. Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary, and Vincent Vidal. Recording and minimizing nogoods from restarts. *JSAT*, 1(3-4):147–167, 2007.
12. A. Ludwig and B. Francyk. COSMA - An Approach for Managing SLAs in Composite Services. In *Proc. of the 6th ICSOC.* Springer Verlag, 2008.
13. O. Martín-Díaz, A. Ruiz-Cortés, A. Durán, and C. Müller. An approach to temporal-aware procurement of web services. In *3rd ICSOC*, pages 170–184, 2005.
14. C. Müller, O. Martín-Díaz, A. Ruiz-Cortés, M. Resinas, and P. Fernández. Improving Temporal-Awareness of WS-Agreement. In *Proc. of the 5th ICSOC*, pages 193–206. Springer Verlag, 2007.
15. C. Müller, A. Ruiz-Cortés, and M. Resinas. An Initial Approach to Explaining SLA Inconsistencies. In *Proc. of the 6th ICSOC.* Springer Verlag, 2008.
16. N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic WS-Agreement Partner Selection. In *15th International WWW Conf., 697–706.* ACM Press, 2006.
17. S. Ouis and M. Tounsi. An explanation-based tools for debugging constraint satisfaction problems. *Applied Soft Computing*, 8(4):1400 – 1406, 2008.
18. O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, and D. Cojocarasu. Managing violations in service level agreements. pages 349–358, 2008.
19. 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.
20. T. Schiex and G. Verfaillie. Nogood recording for static and dynamic constraint satisfaction problems. *Tools with Artificial Intelligence, 1993. TAI '93. Proceedings., Fifth International Conference on*, pages 48–55, 8-11 Nov 1993.
21. E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1995.
22. 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.