Automated Validation of Compensable SLAs

Carlos Müller, Antonio M. Gutierrez, Pablo Fernandez, Octavio Martín-Díaz, Manuel Resinas, and Antonio Ruiz-Cortés

Abstract—A Service Level Agreement (SLA) regulates the provisioning of a service by defining a set of guarantees. Each guarantee sets a Service Level Objective (SLO) on some service metrics, and optionally a compensation that is applied when the SLO is unfulfilled or overfulfilled. Currently, there are software tools and research proposals that use the information about compensations to automate and optimise certain parts of the service management. However, they assume that compensations are well defined, which is too optimistic in some circumstances and can lead to undesirable situations. In this article we discuss about the notion of validity of guarantees with a compensation, which we refer to as compensable guarantees (CG). We describe an abstract model of CGs and we provide a technique that leverages constraint satisfaction problem solvers to automatically validate them. We also present a materialisation of the model of CGs in iAgree, a language to specify SLAs and a tooling support that implements our whole approach. An assessment over 319 CGs taken from 24 real-world SLAs suggests that the expressiveness and effectiveness of our proposal can pave the way for using CGs in a safer and more reliable way.

Index Terms—Analysis, compensation, CSP, penalty, reward, SLA, validation, WS–agreement

1 Introduction

In recent years the use of Service Level Agreements (SLAs) in Information Systems is in continuous rise. They are widely used by the industry in situations where consumers and providers need or desire to explicitly express certain guarantees over the service provisioning. By means of guarantee terms in the SLA [1], one party (herein after the *guarantor*) guarantees to another party (herein after the *beneficiary*) the fulfilment of a Service Level Objective (SLO). For instance, Amazon as provider of its Elastic Compute Cloud Service (AWS EC2) [2] has a term that guarantees its consumers an SLO such that $\text{availability} \geq 99.95\%$. More often than not, guarantee terms of real–world SLAs have associated one or more compensations that represent the consequences of unfulfilling (*penalties*) or overfulfilling (*rewards*) the SLO. For instance, Amazon is penalised with a 10 percent in service credits if the availability of AWS EC2 drops below 99.95 percent. In other scenarios, a provider may be rewarded when the service provided overfulfils the SLO, i.e., providing a better service level than fixed by the SLO fulfilment. In a previous work we coined the concept of Compensable SLAs [3] to refer to SLAs that include at least either a penalty or a reward.

Given the potential economic impact of compensations, it is of the utmost importance to validate that they are well defined in order to avoid undesirable consequences, specially if the action derived from the compensation is automated. For instance, while defining compensations it is desirable to set a limit for the maximum compensation that can be applied if the SLO is not fulfilled. Amazon does so by establishing that they compensate consumers of the AWS EC2 up to a limit of the 30 percent of the monthly bill. Such a limit helps to avoid mistakes like an unbounded, automated penalty that was discarded in 2005 by the UK Royal Mail company after causing a loss of £280 million in one year and a half.[1]

However, despite the importance of checking the validity of compensations, it has not been dealt with by current research proposals that use some form of compensation in their SLAs [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. Instead, they mostly focus on the optimisations of service costs by finding a trade-off between compensation and operation costs or on the automation of several parts of compensation management. These approaches assume that compensations are well defined, which is a too optimistic assumption and the cause of undesirable situations such as the aforementioned UK Royal Mail example.

In this paper, we aim at answering the question "*How can compensations be automatically validated?*" To this end, we build on the compensable SLA model proposed in [3] to provide an automated technique to validate compensable SLAs. Furthermore, we also present a tooling support that implements our validation technique. Our proposal has been evaluated by modelling and analysing the compensations of 24 SLAs of real-world scenarios including 319 guarantee terms. As a result, our technique has proven to be useful for detecting mistakes that are typically derived not only from the manual specification of SLAs in natural language, but also from the complex nature of compensation definitions.

This article is structured as follows: Section 2 introduces two motivating real-world scenarios that are used as running examples in the other sections. In Sections 3 and 4 we present

• *The authors are with the E.T.S. Ingeniería Informática, University of Seville, Sevilla 41012, Spain.*
E-mail: {cmuller, amgutierrez, pablofm, omartindiaz, resinas, aruiz}@us.es.

Manuscript received 9 Feb. 2018; revised 16 Nov. 2018; accepted 3 Dec. 2018. Date of publication 7 Dec. 2018; date of current version 8 Oct. 2021.
(Corresponding author: Carlos Müller.)
Digital Object Identifier no. 10.1109/TSC.2018.2885766

1. Page 3 in http://goo.gl/o7gw6B

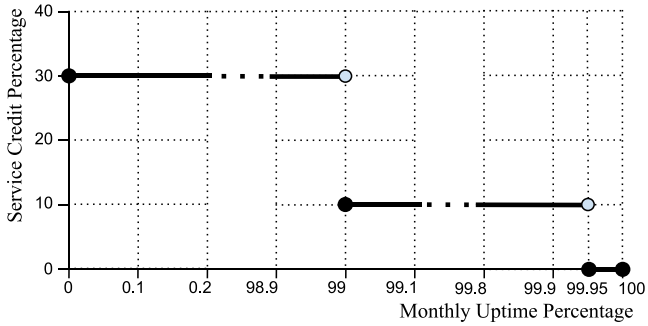

Fig. 1. Example AWS EC2: Penalties for Amazon as provider.

both, the abstract model and validity criteria of the compensation function, and the compensable guarantees, respectively. In Section 5 we explain in detail all issues related to automating the validity checking of compensable guarantees. Our approach is evaluated in Section 6 by analysing real-world compensable SLAs. In Section 7 we analyse the literature to identify related approaches. Finally, in Section 8 we outline some conclusions and future work.

2 Running Examples

In this section two of compensable SLAs found in real world scenarios are introduced as running examples: a computing service and a human-driven IT support service.

2.1 AWS EC2 SLA

Amazon Web Services (AWS) is a service catalogue that has boosted the use of cloud computing in the industry. Amongst the services offered by AWS, the Elastic Compute Cloud (EC2) represents a widely used Infrastructure as a Service (IaaS). The aim of EC2 is to provide a scalable infrastructure to organizations that either have variable needs or need to grow seamlessly without the investment for an internal data center. In this context, the reliability of a virtualised infrastructure represents a key point for IaaS consumers in order to choose a service like AWS EC2.

As a consequence, Amazon has published an SLA for EC2[2] that guarantees the availability of the virtual resources requested by means of the Monthly Uptime Percentage (MUP) service metric. Specifically, it states that MUP will be greater or equal than 99.95 percent. The consequences of not meeting this SLO is defined in two levels: when MUP drops below 99.95 percent and when it drops below 99 percent. Fig. 1 depicts[3] the penalty function [4] of this scenario that is defined as a percentage of discount in the next billing cycle (Service Credit Percentage).

2.2 GNWT SLA

The Government of the Northwest Territories (GNWT) of Canada outsources their IT support. Specifically, the demanded services include issues related to: reporting, user support, problem correction, application enhancement, process and application improvement, and other services. They provide a template for establishing an SLA with an external vendor that provides the mentioned kind of IT support with the desired service levels and penalties and rewards for the parties [20].

Type	Measurement	Penalty
Quarterly Status Report	Delivered at quarterly intervals and not less than five business days before scheduled review meeting	5% of monthly invoice

Example GNWT-1

Severity Code	Initial Response	Estimation Response	Subsequent Responses	Resolution
1	15 minutes	2 hours	Every 30 min.	4 hours

Type	Measurement	Reward	Penalty
Severity 1 Resolution	All Severity 1 problems are resolved in less than 2 hours.	10% of monthly fees	NA
	One or more Severity 1 problems are resolved in over 4 hours.	NA	10% of monthly fees

Example GNWT-2

Type	Measurement	Reward	Penalty
Maximum Problem Aging	No problem is older than 60 days.	5% of monthly fees	NA

Example GNWT-3

Type	Measurement	Reward*	Penalty
Project Delivery	Total elapsed days until delivery is more than 20% greater than planned.	NA	10% of the amount invoiced for the project.
	Total elapsed days until delivery is 20% less than planned.	5% of the amount invoiced for the project.	NA

Example GNWT-4

Fig. 2. Compensation actions extracted from the SLA of GNWT.

Four examples of terms with at least a penalty or a reward have been extracted (cf. Fig. 2) from its SLA template.[4] In example GNWT-1, the government demands the delivery of quarterly reports, which must be received in not less than five days before scheduled review meetings under a penalty of 5 percent of monthly invoice for the provider.

Example GNWT-2 depicts specific times for different milestones that take place in the resolution of problems classified with severity 1: they imply a critical application function unusable or unavailable and no immediate workaround exists. Specifically, an initial response should be received within 15 minutes, an estimation response should be ready in 2 hours, subsequent responses are expected every 30 minutes, and the problem must be resolved within 4 hours. In this case, a reward for the provider applies if all problems are resolved in less than 2 hours, and a penalty for the provider applies if any of them is resolved in more than 4 hours. An additional clause rewarding when no problem is older than 60 days is included in GNWT-3.

Example GNWT-4 includes a term that relates the scheduled project delivery to the real project delivery, which includes a penalty for the provider if the Elapsed Days Percentage until delivery (EDP) is more than 20 percent greater than planned as well as a reward for the provider if the EDP is 20 percent less than planned. Note that, if the latter sentence is taken literally, the reward does not make sense because it would apply if the delivery is exactly 20 percent less than planned and not for an early-delivery that would be more interesting for the government, e.g., 40 percent less than planned. Such a problem can be solved with the automated validation proposed in Section 5.

3 Compensation Functions

A compensation function is defined over a service metric and can represent two different types of compensation: penalty and reward. A penalty represents a compensation from the guarantor to the beneficiary, whereas a reward

2. Available at http://aws.amazon.com/es/ec2/sla/
3. Note that in all functions, a dark point denotes the inclusion of the metric value in the interval and a gray point means the value exclusion.
4. Available at https://goo.gl/m0duhI

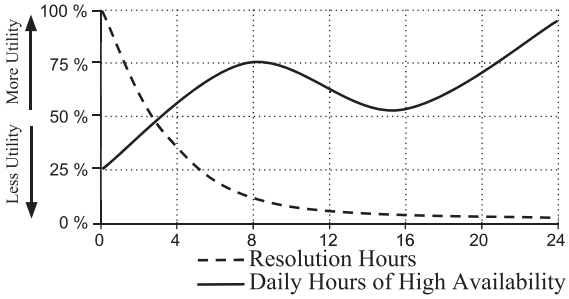

Fig. 3. Example of decreasing or non-monotonic utility functions.

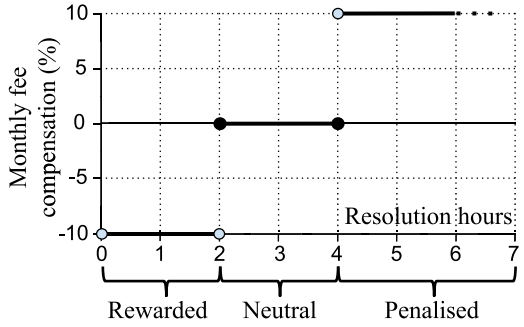

Fig. 4. Compensation function of GNWT-2 ($\mathrm{CF}_{\mathrm{ResolutionHours}}$).

represents a compensation from the beneficiary to the guarantor. Next, we provide a rigorous description.

3.1 Core Definitions

Definition 1 (Metric Values). *Let m be a metric of the service, which assumes values within a domain M_m. The domain can be numerical or categorical, in which case M_m is constituted by a (possibly infinite) set of values $M_m = \{v_1, \ldots, v_n\}$. If it is numerical, it can be either continuous, i.e., $M_m \subseteq \mathbb{R}$, or discrete, i.e., $M_m \subseteq \mathbb{Z}$*

The examples in Section 2 show metrics such as MUP, interventions, and urgent interventions, with bounded domains (e.g., $M_{MUP} = \{v \in \mathbb{R} | 0 \leq v \leq 100\}$); and others such as resolution hours, or EDP with unbounded domains (e.g., $M_{ResolutionHours} = \mathbb{R}_{\geq 0}$).

Definition 2 (Utility Function). *A Utility Function for the metric m, denoted as U_m, is defined as a function from M_m to $\mathbb{R}$ that associates a utility to each of the values; i.e., it defines which metric values in M_m are more interesting for a given party. Utility functions can be defined as decreasing, increasing, constant, or non-monotonic.*

Fig. 3 shows a decreasing utility in the dotted function for the metric *resolution hours* of GNWT-2 example; and a non-monotonic utility function for daily hours of high availability, which aims at optimising different customer goals, such as a common office time of 8 hours per day of high availability, or fully 24 hours per day of high availability, which are the two peeks in the non-dotted function[5]. Utility functions do not appear explicitly in the SLAs because the value each party gives to a service metric is part of their private information and they are usually not willing to share this information with other parties.

Definition 3 (Utility Precedence). *Let U_m a utility function defined on metric m, a precedence relation on M_m, denoted as $\prec$, can be induced in such a way that given any two values $v_1, v_2 \in M_m$ is said that v_1 is less interesting or useful than v_2 when $U_m(v_1) < U_m(v_2)$.*

Dotted function in Fig. 3 represents the utility of the beneficiary of GNWT-2. A utility precedence that could be induced is that the higher value of resolution hours, the less interesting for the beneficiary (e.g., $4 \prec 2$).

Definition 4 (Compensation Function). *A Compensation Function for the metric m, denoted as CF_m, is defined as a function from M_m to a numerical target set C that associates a compensation value to each of the values of m. C can be either continuous, i.e., $C \subseteq \mathbb{R}$, or discrete, i.e., $C \subseteq \mathbb{Z}$. Similarly to utility functions, the compensation functions can be defined as decreasing, increasing, constant, or non-monotonic.*

As a normalised convention that is aligned with related work [4], [22] we establish positive compensations as penalties (that should be compensated from the guarantor to the beneficiary) and negative compensations as rewards (i.e., beneficiary should compensate guarantor).

Fig. 4 shows an example of increasing compensation function taken from the GNWT-2 example. The function denotes: (1) a reward if problems are solved in less than 2 hours; (2) no compensation applies in problems which are solved between 2 and 4 hours, inclusive, and (3) a penalty if the problems are solved in more than 4 hours. Fig. 1 depicts the compensation function of AWS EC2, which does not take negative values, meaning that only penalties are applicable.

Definition 5 (Compensation Regions). *Let CF_m be a compensation function of a given metric m; up to three compensation regions can be defined by such compensation function, namely: penalised, rewarded, and neutral.*

$$\mathrm{Penalised}(\mathrm{CF_m}) = \{v_i \in \mathrm{M_m} | \mathrm{CF_m}(v_i) > 0\}$$
$$\mathrm{Neutral}(\mathrm{CF_m}) = \{v_i \in \mathrm{M_m} | \mathrm{CF_m}(v_i) = 0\}$$
$$\mathrm{Rewarded}(\mathrm{CF_m}) = \{v_i \in \mathrm{M_m} | \mathrm{CF_m}(v_i) < 0\}.$$

Fig. 4 shows these three potential subsets in GNWT-2 example. Thus, $\forall v_i < 2$ in the Figure, v_i is a rewarded value; $\forall v_i > 4$ in the Figure, v_i is a penalised value; and $\forall v_i | 2 \leq v_i \leq 4$ in the Figure, v_i is a neutral value.

3.2 Validity Criteria of Compensation Functions

Our proposal to define the validity criteria of a Compensation Function relies on the notions of Consistency and Saturability, that we rigorously describe in the following properties.

Property 1 (Consistency). *Let be U_m a utility function for m defined by the beneficiary, a compensation function CF_m is said to be consistent w.r.t. U_m, denoted as $Consistent(CF_m, U_m)$, if the compensation for a less interesting value of the metric is greater than or equal to the compensation for a more interesting value, according to the utility precedence defined by U_m, i.e., if it holds:*

$$\forall v_1, v_2 \in \mathrm{M_m} \cdot v_1 \preceq v_2 \Rightarrow \mathrm{CF_m}(v_1) \geq \mathrm{CF_m}(v_2).$$

5. Example called "horizontal demand" in [21].

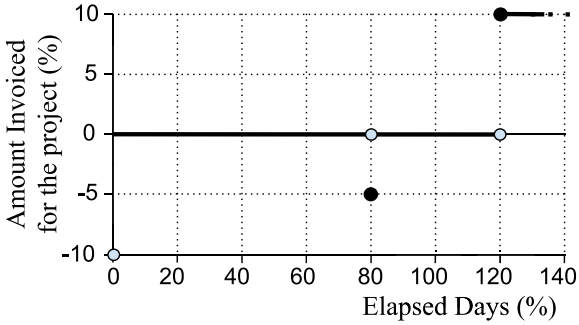

Fig. 5. Example of inconsistent CF of GNWT-4 ($\mathrm{CF_{EDP}}$).

Let us analyse the consistency of the compensation functions of AWS EC2 and GNWT-4 examples assuming that, in AWS EC2, the beneficiary establishes that the higher the metric value, the more interesting, i.e., the utility function $\mathrm{U_{MUP}}$ is monotonically increasing, while in GNWT-4, the lesser the metric value, the less interesting, i.e., the utility function $\mathrm{U_{EDP}}$ is monotonically decreasing. In the case of AWS EC2, the decreasing $\mathrm{CF_{MUP}}$ depicted in Fig. 1 is consistent w.r.t. the monotonically increasing $\mathrm{U_{MUP}}$. As counterexample, the non-monotonically increasing $\mathrm{CF_{EDP}}$ depicted in Fig. 5 is not consistent w.r.t. the $\mathrm{U_{EDP}}$ in the GNWT-4 example because an early-delivery in the 40 percent of EDP, that is a less interesting value for the guarantor, implies a higher compensation than 80 percent of EDP.

Property 2 (Saturability). *A compensation function CF_m is said to be saturated w.r.t. a threshold $\tau = (\tau_{min}, \tau_{max})$, where $\tau_{min}, \tau_{max} \in \mathbb{R}$, denoted as $Saturated(CF_m, \tau)$ if the threshold delimits the higher compensation, either penalty or reward, i.e., if the following holds:*

$$\forall v_i \in \mathrm{M_m}, \mathrm{CF_m}(v_i) \in [\tau_{min}, \tau_{max}].$$

This property prevents the definition of unbounded compensations, where the boundary is defined by a threshold τ. One may think that it would be enough to check that compensations do not grow infinitely. In practice, however, this is often not enough because, although bounded, it does not guarantee that the compensation is reasonable according to the problem domain. For instance, a compensation that grows linearly with the number of minutes of unavailability in a month is bounded because the number of minutes in a month is also bounded. However, in practical terms, such a compensation may be unreasonable for the provider. Therefore, we define the threshold as a way to set a reasonable boundary in a domain-specific manner. Since compensation functions of the running examples are defined as piecewise functions, they are saturated by definition.

Property 3 (Validity). *Let U_m be a utility function defined for m by the beneficiary, and let τ be a threshold for m, a compensation function CF_m is said to be valid, denoted as $\mathrm{Valid(CF_m, U_m, \tau)}$, if it is consistent w.r.t. U_m and saturated w.r.t. τ. i.e., if the following holds:*

$$\mathrm{Consistent(CF_m, U_m)} \wedge \mathrm{Saturated(CF_m, \tau)}.$$

According to this definition, the compensation functions of Figs. 1 and 4 are valid. On the contrary, the compensation function of GNWT-4 example, depicted in Fig. 5, is not consistent and therefore, not valid.

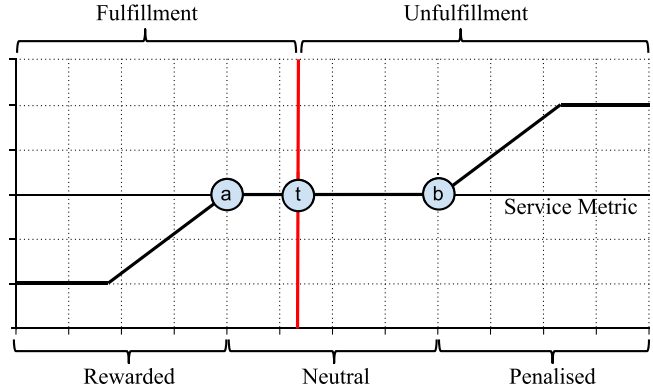

Fig. 6. A generic example of compensable guarantee showing the fulfilment and compensation regions.

4 Compensable Guarantees

A guarantee term guarantees the fulfilment of a certain SLO to a beneficiary (e.g., $\mathrm{ResolutionHours} \leq 4\mathrm{h}$). When a guarantee term also includes penalties and/or rewards to compensate the unfulfilment and/or overfulfilment of the SLO, it becomes a compensable guarantee term and by extension its containing SLA becomes a compensable SLA [3]. Next, we provide some core definitions to formally define the validity of compensable guarantees.

4.1 Core Definitions

Definition 6 (Service Level Objective). *Let m be a service metric, SLO_m is said to be Service Level Objective over a metric m, if it represents a predicate over m.*

Examples of valid SLOs are $\mathrm{ResolutionHours} \leq 4\mathrm{h}$, and $\mathrm{MUP} \geq 99.95\%$, in GNWT-2 and AWS, respectively.

Definition 7 (Fulfilment Regions). *Given an SLO_m two regions over the values of m can be defined, namely fulfilment and unfulfilment.*

$$\mathrm{Fulfilment}(SLO_m) = \{v_i \in M_m | SLO_m(v_i)\}$$
$$\mathrm{Unfulfilment}(SLO_m) = \{v_i \in M_m | \neg SLO_m(v_i)\}.$$

According to this definition, the fulfilment region of GNWT-2 scenario is $\mathrm{ResolutionHours} \leq 4\mathrm{h}$ and the unfulfilment region is $\mathrm{ResolutionHours} > 4\mathrm{h}$.

Definition 8 (Compensable Guarantee). *A compensable guarantee defined over a metric m, denoted as CG_m, is a tuple of the form (SLO_m, CF_m), where SLO_m is a service level objective and CF_m is a compensation function, both defined over the same service metric m. $CG_m.SLO$ refers to the SLO of CG_m and $CG_m.CF$ to the compensation function of CG_m.*

According to this definition, all guarantees in the article are compensable guarantees. Fig. 6 shows a typical compensation function that depicts the relationships between the fulfilment regions delimited by the SLO and the compensation regions defined by the compensation function (cf. Section 3.1). Moreover, as shown in this Figure, it is important to highlight that fulfilment regions are not

necessarily coupled with compensation regions. Specifically, the Figure exemplifies a case in which metric values between t and b are unfulfilled but not penalised, and similarly metric values between a and t are fulfilled but not rewarded.

4.2 Validity Criteria of Compensable Guarantees

The validity of a compensable guarantee resides not only in the validity of its compensation function, but also in the existing coherence between the compensation function and the SLO. Considering the coherence as a property, it can be defined as follows.

Property 4 (Coherence). *A compensable guarantee CG_m is said to be coherent, denoted as $Coherent(CG_m)$, if it does the fulfilment regions w.r.t. compensation regions, which implies a threefold condition namely, there is no unfulfilled value that is rewarded, no fulfilled value that is penalised, and there is at least one fulfilled value that is neutral, i.e., if the following holds:*

$$\begin{aligned}&\mathrm{Unfulfilment}(\mathrm{CG_m.SLO}) \cap\ \mathrm{Rewarded}(\mathrm{CG_m.CF}) = \varnothing\\ &\wedge\ \mathrm{Fulfilment}(\mathrm{CG_m.SLO}) \cap \mathrm{Penalised}(\mathrm{CG_m.CF}) = \varnothing\\ &\wedge\ \mathrm{Fulfilment}(\mathrm{CG_m.SLO}) \cap \mathrm{Neutral}(\mathrm{CG_m.CF}) \neq \varnothing.\end{aligned}$$

According to this property, GNWT-2 compensable guarantee of Fig. 4 is coherent with its SLO (shown in Fig. 2 as $\mathrm{ResolutionHours} \leq 4$) due to all resolution hour in the SLO fulfilment region is in either rewarded region when the SLO is overfulfilled or in the neutral region; and all resolution hour in the unfulfilment region (i.e., $\mathrm{ResolutionHours} > 4$) is in the penalised region.

Based on Properties 3 and 4 we can formalise the validity of a compensable guarantee as follows:

Property 5 (Validity of a CG). *Let U be a utility function defined for metric m by the beneficiary, and let τ be a threshold for m, a compensable guarantee CG_m is said to be valid, denoted as $\mathrm{Valid}(\mathrm{CG_m}, \mathrm{U_m}, \tau)$, if it is coherent, i.e., the SLO and CF are coherent between themselves; and it contains a valid compensation function w.r.t. U_m and τ.*

$$\mathrm{Coherent}(\mathrm{CG_m}) \wedge \mathrm{Valid}(\mathrm{CG_m.CF}, \mathrm{U_m}, \tau).$$

According to this property, GNWT-2 compensable guarantee of Fig. 4 is valid because it is coherent and its compensation function is valid.

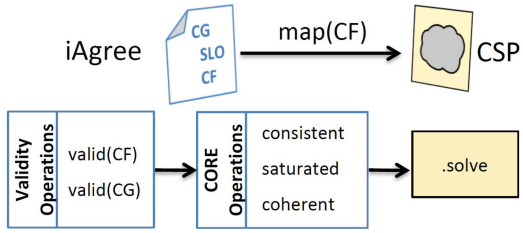

Fig. 7. Process to automate the validity checking.

5 Validity Checking of Compensable Guarantees

Checking whether a compensable guarantee contains any errors become a tedious and time-consuming task, and it is error-prone in itself. Our approach to automate this process is depicted in Fig. 7 and requires a language to specify compensable guarantees. Thus, we propose the use of *iAgree* [23], a WS–Agreement-based [1] language with precise semantics (cf. Section 5.2). The underlying idea is to leverage off-the-shelf Constraints Satisfaction Problem (CSP) reasoners to automate the analysis of operations; more specifically, to extract information from the CSPs that represents the compensation function of the guarantee specified in iAgree. As shown in Fig. 7, we establish a hierarchy of operations: the $\mathrm{Validity}$ properties formalised in Sections 3 and 4 (included in the reduced glossary of Table 1) for CF (Property 3) and CG (Property 5), respectively, are derived from three core operations, namely: $\mathrm{Consistency}$ (Property 1), $\mathrm{Saturability}$ (Property 2), and $\mathrm{Coherence}$ (Property 4). Each core operation can be

TABLE 1
Definitions and Properties Glossary

Definition	Reduced Explanation
$\mathrm{M_m}$	Set of all values for a metric m.
$\mathrm{U_m}$	Function that associates a utility to each value of m.
Precedence $\prec$	Precedence relation denoting that a value of m is less interesting than other.
$\mathrm{CF_m}$	Function that associates a compensation to each of the values of a metric.
Compensation Regions	$\mathrm{Penalised}(\mathrm{CF_m})$, $\mathrm{Rewarded}(\mathrm{CF_m})$, $\mathrm{Neutral}(\mathrm{CF_m})$; where the metrics values have an associated penalty, reward, or none, respectively.
$\mathrm{SLO_m}$	A service level over m (i.e., a predicate over m).
Fulfilment Regions	$\mathrm{Fulfilment}(\mathrm{SLO_m})$, $\mathrm{Unfulfilment}(\mathrm{SLO_m})$; where the metrics values hold and do not hold the SLO predicate, respectively.
CG	A guarantee that includes either a penalty, a reward, or both. It comprises both, a $\mathrm{SLO_m}$ and a $\mathrm{CF_m}$.
Property	**Reduced Explanation**
$\mathrm{Consistent}(\mathrm{CF_m},\mathrm{U_m})$	A CF is consistent if the compensation for a less interesting value of m is greater than or equal to the compensation for a more interesting value, according to the utility precedence.
$\mathrm{Saturated}(\mathrm{CF_m}, \tau)$	A CF whose penalties or rewards are always less than a threshold (τ).
$\mathrm{Valid}(\mathrm{CF_m}, \mathrm{U_m}, \tau)$	A CF that is consistent and saturated.
$\mathrm{Coherent}(\mathrm{CG})$	A CG whose fulfilment and compensation regions make sense (e.g., there is no unfulfilled value that is rewarded, etc).
$\mathrm{Valid}(\mathrm{CG_m}, \mathrm{U_m}, \tau)$	A CG that is coherent and its CF is valid.

```
1  GNWT_SLA:
2  ...
3  context:
4    provider: Provider
5    consumer: Northwest Territories Government
6    validity:
7      initial: '2016-07-13T00:00:00.000Z'
8    definitions:
9      schemas:
10       PIP:
11         description: Compensation for next monthly bill
12         type: integer; unit: '%'
13         minimum: -100; maximum: 100
14 terms:
15   metrics:
16     EDP: "m"
17       schema:
18         description: elapsed days until delivery (%)
19         type: integer; unit: '%'
20         minimum: -200; maximum: 200
21   guarantees:
22     id: GNWT-4
23      of:
24        objective: EDP < 120
25        window:
26          type: static
27          period: monthly
28          initial: '2016-07-13T00:00:00.000Z'
29        penalties:
30          over: PIP
31          of:
32            value: 10
33            condition: EDP ≥ 120
34        rewards:
35          over: PIP
36          of:
37            value: -5
38            condition: EDP == 80
```

Fig. 8. GNWT-4 compensable guarantee term in *iAgree* syntax.

expressed in terms of a standard CSP-based reasoning operation (cf. *.solve* in Fig. 7). This scheme requires to define a mapping between iAgree compensations and CSP denoted as *map (CF)* in this Figure.

5.1 Inferring Utility Function and Saturability Thresholds

As described in Section 3.2, the validity of a compensation function requires knowing besides the compensation function of each compensable guarantee, both, (i) the utility function of the beneficiary for each metric as far of the $\mathrm{Consistency}$ property; and (ii) the threshold for the $\mathrm{Saturability}$ property. However, these elements are not usually public, and hence, they do not appear in the SLA. Therefore, they must be provided by domain experts during the validity checking. However, although the utility function is not explicitly described in the SLA, its structure can be roughly estimated from the SLO; based on our analysis, this assumption has proved to be useful in order to avoid an explicit utility function definition in most of SLAs found in the industry.

After analysing up to 24 SLAs, 95 percent of their 319 compensable guarantees include simple SLOs in the form of $\mathrm{metric} > \mathrm{target\_value}$ or $\mathrm{metric} < \mathrm{target\_value}$. Therefore, the intuitive related utility can be described as a monotonic linear function directly proportional (a greater value of the metric is more useful) or inversely proportional (a lower metric value is more useful) to metric values, respectively. For instance, in the SLO: $\mathrm{availability} \geq 99$, the utility function can be described as $\mathrm{U_{availability}(x) = x}$, and in the SLO $\mathrm{EDP} \leq 120$ (cf. SLO of Fig. 8), the utility function can be described as $\mathrm{U_{EDP}(x) = -x}$. Therefore, to avoid manual modelling, we apply this simple automatic criteria using the SLO inequation to define the utility function in our tooling. This approach is not valid when there is no explicit SLO, the SLO is specified as an equation instead of an inequation, or the SLO is more complex, but it covers the 95 percent of modelled agreements. In cases where the SLO is defined as an equation (e.g., $\mathrm{UnavailableMinutes} = = 0$), the domain expert had to model it, with no side-effects in validation, to the SLO $\mathrm{UnavailableMinutes} \leq 0$, that allows to infer a proper utility function for the metric (the sign of the inequation can be decided based on the metric semantic). This modelling decision is valid in all the analysed scenarios but should be extended for objectives such as $\mathrm{deliveryDay} = \mathrm{Friday}$; in such cases, in spite formalisation provides support for that, the tools developed would need to be extended since it is required for automatically inferring a utility function. As a potential alternative, if the utility function was provided by the user, then SLOs could be defined as an equation.

Regarding the saturability property, there are different situations where the maximum (τ_{max}) and the minimum (τ_{min}) values are defined in the SLA. In some cases, as in Fig. 8, compensation is defined through constant values, 10 and -5, in lines 32 and 37, so the maximum and minimum possible values are in fact saturability thresholds. In cases where there are no explicit thresholds in the SLA and they are not included in analysis (e.g.,: $\mathrm{c} = 100 \times \mathrm{DelayDays}$), the default approach is using the domain of the compensation variable as threshold. Thus, we consider that if the compensation can reach maximum or minimum domain value, it is not saturated.

5.2 iAgree as Specification Language

The validity properties identified for compensable guarantees deal with four key elements, namely: service metrics (m), compensable guarantees (CG), service level objectives ($SLOs$), and compensation functions (CF). In such a context, iAgree already supported with a precise semantics m and $SLOs$, complementary, as a key contribution of this paper, we extend the support to include CG by means of a formal semantics for CF; this extension is detailed in Section 5.3.

Fig. 8 depicts an excerpt of the GNWT SLA in iAgree including the GNWT-4 compensable guarantee. Specifically, an iAgree SLA includes two types of elements, namely: a *context* and a set of *terms*. The context specifies who the consumer and provider are and defines *schemas* that specify the domain and unit for the compensations used in the SLA (we refer to the compensation value as *c* in the following sections; cf. Penalty Invoice Percent, PIP in the example, line 10). The terms are divided into *metrics* and *guarantees*. Metrics are specified by means of their schemas, which define their values (m; cf. EDP in the example, line 16). Therefore, schemas can be seen as a function domain that returns the domain of either metrics or compensation defined as $\mathrm{domain}(m) = \mathrm{M_m}$ and $\mathrm{domain}(c) = \mathrm{M_c}$, respectively (e.g., $\mathrm{domain(EDP)} = [-200, \ldots, 200]$, line 20).

Guarantees include both: (i) the SLO defined on a service metric ($\mathrm{SLO_m}$; cf. $\mathrm{EDP} < 120$ in the example, line 24), and (ii) the Compensation Function (CF_m). Following the WS–Agreement specification, $\mathrm{CF_m}$ is defined by specifying *penalties* and *rewards* separately. Neutral values are not explicitly specified and include those that are not part of any penalty or reward. For each penalty and reward, the compensation domain is defined by means of element *over*.

TABLE 2
Compensation Function to CSP

CSP of compensation function - $map(CF_m)$	
iAgree Element	CSP Mapping
definitions: schemas: c	$V \leftarrow V \cup c$ $D \leftarrow D \cup domain(c)$
terms: metrics: m	*[-for CG Coherence-]* $V \leftarrow V \cup m$ $D \leftarrow D \cup domain(m)$
penalties: over: c of: value: $P_{val,i}$ condition: $P_{cond,i}$	$C \leftarrow C \cup P_{cond,i} \implies c == P_{val,i}$
rewards: over: c of: value: $R_{val,j}$ condition: $R_{cond,j}$	$C \leftarrow C \cup R_{cond,j} \implies c == R_{val,j}$
[implicit neutral region]	$C \leftarrow C \cup NOT(P_{cond,i}\ OR\ R_{cond,j}) \implies c == 0$

The definition of the penalty (or reward) part of the compensation function is done as a piecewise function through a set of $(value, condition)$ pairs (e.g., '10', EDP $\geq$ 120, lines 32 and 33 in the penalty of the example). According to our definition of $\mathrm{CF_m}$, both penalty and reward are defined over the same schema, a penalty schema in our example. It is important to highlight that, based on the analysis of SLAs in the industry (cf. Section 6) two assumptions can be made in our example: first, following our CF definition we use the PIP as the compensation metric assuming a positive value as penalties and negative value as rewards; and second, the usage of piecewise functions proves to be sufficiently expressive to model a wide range of compensations functions in real-world scenarios as shown in Section 6.

5.3 Formal Semantics of iAgree Compensation Functions

The primary objective of formalising iAgree compensation functions is to establish a sound basis for an automated support. Therefore, according to the formalisation principles defined by Hofstede et al. [24], the style and target domain should be chosen accordingly (*Primary Goal Principle*). In our case, we follow a transformational style and we propose CSP as target domain because of two major reasons. First, compensations are specified by means of constraints of their related metrics. Thus, a quite straightforward way to implement the checking of the introduced properties (cf. Sections 3 and 4). Second, similar techniques have been previously applied by the authors [25], [26] to analyse several SLA problems, namely: (i) the conflicts between agreement terms (e.g., to avoid inconsistencies); (ii) the SLA violation (e.g., to detect violations and explain potential causes); and (iii) the compliance between agreements and agreement templates.

A CSP is defined as a triple (V, D, C) of a set of variables V, their domains D and a number of constraints C. A solution of a CSP is an assignment of values to the variables in V from their domains in D so that all the constraints in C are fulfilled. A CSP operation that returns if it has at least one solution is called solve, and there are a number of solvers that automates this CSP operation. Our approach, thus, involves mapping the iAgree CF to a CSP and then using the solve operation of a CSP solver to check the different properties. Next we explain this mapping in detail, summarising it in Table 2.

In the following, to refer to the elements in the iAgree document we use the following notation. For each compensable guarantee, CG, in an iAgree document, we will use CG.SLO, CG.CF to refer to its SLO and compensation function, respectively. The compensation function is defined with penalties and rewards, notated as P and R, respectively. Furthermore, m and c refer to the metric and the compensation variable used in the guarantee. Similarly, and abusing the notation, for each set of penalties that conform the piecewise penalty function P (resp. rewards R), $P_{cond,n}$, and $P_{val,n}$ refer to the n-th condition and the n-th value of the penalty, respectively (resp. $R_{cond,n}$, $R_{val,n}$ for the set of rewards R).

Mapping the Compensation Function (CF): Table 2 describes the CSP of a compensation function defined as a set of penalties and rewards. The compensation and the metrics (together with their schemas) in Agree are the variables (and their domains) in this CSP.

Each $(\mathrm{value}, \mathrm{condition})$ pair in penalties or rewards in iAgree is mapped to a constraint of the form $\mathrm{condition} \Rightarrow variable == value$. Hereinafter, we use c to refer to the compensation variable (penalties and rewards could refer to different compensation variables but then, they would be analysed as different compensation functions). By definition in iAgree, the compensation is zero outside the defined penalty and reward expressions so we add an additional constraint with this consideration. Finally, all these constraints are combined with an AND operator. Note that this mapping corresponds to a well-defined piecewise function; as a potential extension, before the presented compensation analysis, we could check whether the compensation is a mathematical well-defined function (e.g., two different penalty values with the same condition cannot exist).

The resulting CSP of mapping the compensation function of guarantee GNWT-4 in Figure 8 is as follows:

$$
\begin{aligned}
&\mathrm{map}(\mathrm{CF_{EDP}})\\
&\quad = \{\{\mathrm{EDP}, \mathrm{PIP}\}, \{\mathrm{int}[-200..200], \mathrm{int}[-100..100]\},\\
&\qquad \{(\mathrm{EDP} \geq 120 \Rightarrow \mathrm{PIP} == 10)\ \mathrm{AND}\\
&\qquad (\mathrm{EDP} == 80 \Rightarrow \mathrm{PIP} == -5)\ \mathrm{AND}\\
&\qquad \mathrm{NOT}(\mathrm{EDP} == 80\ \mathrm{OR}\ \mathrm{EDP} \geq 120) \Rightarrow (\mathrm{PIP} == 0)\}\}.
\end{aligned}
$$

This CSP is used to evaluate the analysis operations for the properties of CF and CG, which are described next.

5.4 Automated Checking Operations

In this section we explain the automated technique to check the core operations of the proposed properties (cf. Table 3). Note that we stand by: (i) the CF mapping in Table 2 ($\mathrm{map}(\mathrm{CF_m})$), (ii) the inferred utility function ($\mathrm{U_m}$), and (iii) the inferred saturability thresholds (τ_{min}, τ_{max}). Since core operations are used to check the $\mathrm{Validity}$ operations, we have also included them to provide all checking operations.

TABLE 3
Automated Core Operations Using Solve Operation of CSPs

Property 1: Consistency
$\text{Consistent}(\text{CF}_\text{m}, \text{U}_\text{m}) \iff$ $\text{NOT}(\text{solve}(\text{map}(\rho_{\text{m}_1,\text{c}_1/\text{m,c}}(\text{CF}_\text{m}))$ $\text{AND map}(\rho_{\text{m}_2,\text{c}_2/\text{m,c}}(\text{CF}_\text{m}))\ \text{AND}$ $(\text{c}_1 > \text{c}_2)\ \text{AND}\ (\text{U}_{\text{m}_1} > \text{U}_{\text{m}_2})))$
Property 2: Saturability
$\text{Saturated}(\text{CF}_\text{m}, \tau) \iff$ $\text{NOT}(\text{solve}(\text{map}(\text{CF}_\text{m})\ \text{AND}\ (\text{c} > \tau_{\max}\ \text{OR}\ \text{c} < \tau_{\min})))$
Property 4: Coherence
$\text{Coherent}(\text{CG}_\text{m}) \iff$ $\text{solve}(\text{map}(\text{CG}_\text{m}.\text{CF})\ \text{AND}\ (\text{c} == 0\ \text{AND}\ \text{CG}_\text{m}.\text{SLO}))$ $\text{AND NOT}(\text{solve}(\text{map}(\text{CG}_\text{m}.\text{CF})\ \text{AND}$ $(\text{c} > 0\ \text{AND}\ \text{CG}_\text{m}.\text{SLO})\ \text{OR}$ $(\text{c} < 0\ \text{AND NOT}(\text{CG}_\text{m}.\text{SLO}))))$

Operation for Consistency *(Property 1).* Since CSPs can only solve satisfiability problems (i.e., exists a solution) and we have to check if this property holds for all possible values, we verify this property with the solve operation and verifying that there is not value which solves the *inconsistency* of the compensation function. That is, there do not exist two different metric values, m_1 and m_2, where m_1 has higher utility than m_2 ($\text{U}_{\text{m}1}$ and $\text{U}_{\text{m}2}$, respectively), but also higher compensation value (c_1 and c_2, respectively). For instance, if the CF is defined on metric EDP, we define two different metrics, EDP1 and EDP2, with the same compensation function (that is duplicating the penalty and rewards conditions for the duplicated metrics). We express this using $\rho_{\text{a/b}}(\text{CF})$, which means that we rename b *with* a *in CF*. The property is evaluated by adding a constraint that relates the two compensations and utilities for these variables so that one variable can have both a higher compensation (i.e., higher penalty) and a higher utility than the other. If the resulting constraints can be solved (i.e., it has a solution) it implies that the function is not consistent. Otherwise, it would mean that there are two values of the metric variable for which one have both higher penalty and higher utility than the other.

In our example, a greater delay in delivery is less useful for the customer, this means that the utility function is inversely proportional to the metric EDP. Thus, according to our described utility inference, a possible utility function U could be the negative function of the metric: $\text{U}_\text{EDP} = -\text{EDP}$, and consequently, the constraint $(\text{U}_{\text{m}_1} > \text{U}_{\text{m}_2})$ replaced by $(-\text{EDP}_1 > -\text{EDP}_2)$ in our example.

$$\begin{aligned}&\text{Consistent}(\text{CF}_\text{EDP}, \text{U}_\text{EDP}) \Leftrightarrow \\ &\quad \text{NOT}(\ \text{solve}(\text{map}(\rho_{\text{EDP}_1,\text{PIP}_1/\text{EDP,PIP}}(\text{CF}_\text{EDP})) \\ &\quad \text{AND map}(\rho_{\text{EDP}_2,\text{PIP}_2/\text{EDP,PIP}}(\text{CF}_\text{EDP})) \\ &\quad \text{AND}\ (\text{PIP}_1 > \text{PIP}_2)\ \text{AND}\ (-\text{EDP}_1 > -\text{EDP}_2))),\end{aligned}$$

where ρ_1 (resp. ρ_2) renames EDP and PIP by EDP_1 and PIP_1 (resp. EDP_2 and PIP_2).

As this CSP is satisfiable, i.e., solve returns a solution, the example compensation function is *inconsistent*. This is an effect of a wrong compensation definition, which only rewards when the value for EDP is exactly 80, but does not reward more useful values. For instance, the reward does not apply if a project is delivered one month earlier than planned; but it only applies if it is delivered exactly one week earlier than planned.

Operation for Saturability *(Property 2).* For this property, the operation receives two additional input values, the saturability thresholds, so we only check if the metrics are outside these two values. If they are, then we consider it not saturated. In our example, following a domain expert consideration, 30 and -30 are suitable candidate for saturability thresholds. Thus, the CSP for checking the saturation in the example compensation function is:

$$\begin{aligned}&\text{Saturated}(\text{CF}_\text{EDP}, 30, -30) \Leftrightarrow \\ &\text{NOT}(\ \text{solve}(\text{map}(\text{CF}_\text{EDP})\text{AND}(\text{PIP} > 30\ \text{OR}\ \text{PIP} < -30))).\end{aligned}$$

As the compensation variable cannot reach maximum or minimum values in any case (its only possible values are 10, 0 and -5), the problem is not satisfiable and the compensation is therefore *saturated*.

Operation for Validity *of Compensation Function (Property 3).* This is a derived operation of core *consistent* and *saturated* operations.

$$\begin{aligned}&\text{Valid}(\text{CF}_\text{EDP}, \text{U}_\text{EDP}, 30, -30) \Leftrightarrow \\ &\text{Consistent}(\text{CF}_\text{EDP}, \text{U}_\text{EDP})\text{AND Saturated}(\text{CF}_\text{EDP}, 30, -30).\end{aligned}$$

In our example, the compensation is *invalid* (it is *saturated* but not *consistent*).

Operation for Coherence *(Property 4).* This property involves checking one existentially quantified constraint (it exists at least one value that is neutral and fulfilled), and one universally quantified constraint (no value is fulfilled and penalised or unfulfilled and rewarded at the same time). Therefore, we build and solve two different constraints to evaluate this property. First, if it exists a neutral compensation region for an SLO fulfilled value, and if it does not exist a positive compensation value (penalty) when the SLO is fulfilled or a negative compensation value (reward) when SLO is unfulfilled. The operation to check the coherence between compensation function and the SLO in our example is:

$$\begin{aligned}&\text{Coherent}(\text{CG}_\text{EDP}) \Leftrightarrow \text{solve}(\text{map}(\text{CG}_\text{EDP}.\text{CF})\ \text{AND} \\ &\quad (\text{PIP} == 0\ \text{AND}\ \text{EDP} < 120)) \\ &\text{AND NOT}(\text{solve}(\text{map}(\text{CG}_\text{EDP}.\text{CF})\ \text{AND} \\ &\quad ((\text{PIP} > 0\ \text{AND}\ \text{EDP} < 120) \\ &\quad \text{OR}\ (\text{PIP} < 0\ \text{AND NOT}(\text{EDP} < 120))))).\end{aligned}$$

As it exists a neutral compensation when EDP is less than 120 and it does not exist any positive compensation when EDP is less than 120 neither negative when EDP is greater or equal than 120, the compensable guarantee is *coherent*.

Operation for Validity *of Compensable Guarantee (Property 5).* The compensable guarantee is valid if it is coherent with compensation function (Property 4) and the compensation function is valid (Property 3). Therefore we provide this operation as a derived operation of the core *coherent* operation and the derived *valid compensation* operation.

$$\begin{aligned}&\text{Valid}(\text{CG}_\text{EDP}, \text{U}_\text{EDP}, 30, -30) \Leftrightarrow \\ &\quad \text{Coherent}(\text{CG}_\text{EDP})\ \text{AND Valid}(\text{CG}_\text{EDP}.\text{CF}, \text{U}_\text{EDP}, 30, -30).\end{aligned}$$

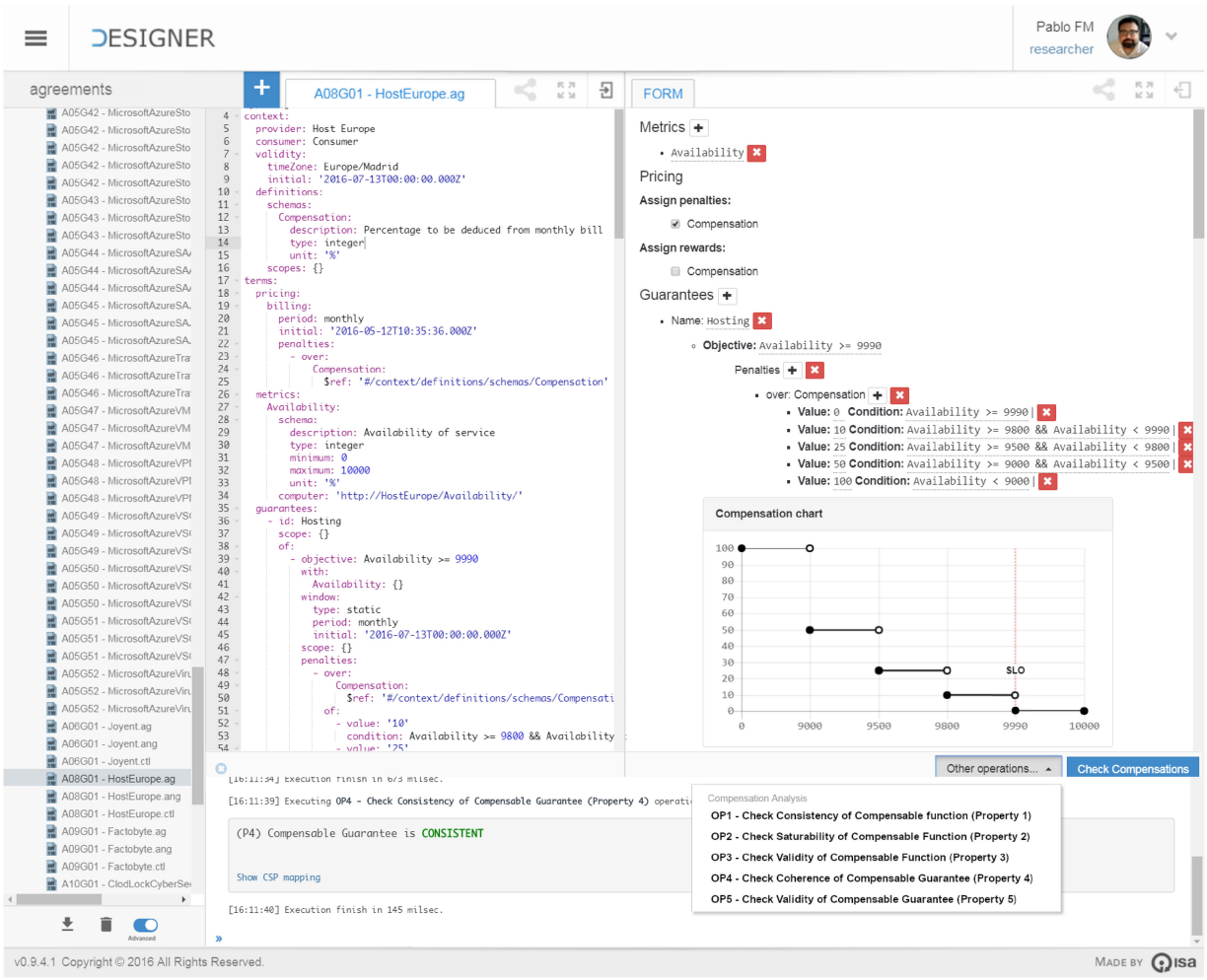

Fig. 9. Screenshot of our designer studio tool.

For our example, the compensation function is *invalid*, hence the compensable guarantee is also *invalid*.

5.5 Tooling Support

In order to assist in the modelling and analysis of properties, an ecosystem of tools has been developed[6] that is composed of (i) an analysis framework, that is implemented using a MiniZinc [27] CSP solver; and (ii) a designer tool that are integrated by means of REST APIs.

As shown in Fig. 9, the designer studio (part of the larger framework Governify) provides a GUI that allows modelling and analyzing compensable SLAs: specifically, it is possible to generate workspaces (far left section of the tool) with agreements written in iAgree with a YAML/JSON format (left editor) or using a form that renders the compensation function in a graphical way (right editor). Once the agreement is modelled, a general validity check of compensations is available (bottom right blue button) or an individual operation check for each property identified in Sections 3 and 4 can be invoked (bottom right gray dropdown). The result of the operations is shown in the console of the tool (bottom section) highlighting the logic (true/false) outcome of the operation. In case of properties P1, P2 and P4 a CSP mapping can be shown (blue link in the console) describing the actual MiniZinc CSP that is executed in the engine to solve the property check.

Following the micro-service architecture principles, the analysis module exposes a well defined REST API[7] that is divided into two main resources: first, the models resource, that corresponds to the different types of documents allowed as inputs for the analysis (e.g., agreements) and their serialisation formats (currently JSON and YAML). Second, the operations resource, that allocates the different analysis available for a given model. From an internal perspective, the agreement module develops a two stage model transformation process: first, the agreement model is transformed using the mapping rules presented in Section 5.4 into a simplified CSP model composed of variables, domains and constraints; second, the CSP model is serialised into the MiniZinc CSP language. In order to execute the solver, a docker container has been integrated so MiniZinc engine can be invoked dynamically for each file generated.

6 Evaluation in Real-World Scenarios

In this section we describe how we have evaluated our proposal. In particular, the goal of the evaluation was to answer the following research questions:

RQ1: How Expressive is our Compensations Model in Comparison to Real-World SLAs? We want to know whether

6. Available at https://isa-group.github.io/2017-12-compensations/

7. A portal for this API is available at https://goo.gl/zbCG9i

TABLE 4
Summary on the Expressiveness and Effectiveness of Our Proposal in Real-World Compensable SLAs

SLA Features	A01-GNWT	A02-Amazon	A03-Rackspace	A07-Oregon DOT	A14-Verizon	A16-Sandetel	A17-Gehronte	A18-Kerala	A19-GSA	A20-India Gov.	A21-RENFE	A22-DHL	A23-Movistar	A24-Baroda Bank	A12-Cdmon	A04-OVH, A06-Joyent A15-GoGrid	A05-Microsoft, A08-HostEurope A09-Factobyte, A10-CloudLock A11-Cisco, A13-Google
Number of compensable guarantees	23	7	16	4	31	5	38	8	19	30	3	1	2	2	1	7	122
Number of metrics involved in CF	1	1	1	1	1	1	1 / 2+	1 / 2+	1	1	1	1	1	2+	1	1	1
CF does only involve the SLO metric	19	5	7		9	5		6	2						1		122
CF involves an aggregation of SLO metric	2									30							
CF involves the SLO metric multiplied by a constant value	2	2	9		22		37	1	1					1		7	
CF involves other service											3	1					
CF involves metrics values from more than one billing cycle													2				
CF denotes whether the SLO is fulfilled during more than one billing cycle				4			1	1	16					1			
CF is not consistent	2				5												
CF is not saturated		1	1	4	1										1		
CG is not coherent						2											

the compensation model that we use is expressive enough to model a wide variety of real-world SLAs and which are the characteristics of the SLAs that we are not able to express.

RQ2: Are the compensations in real-world SLAs valid according to our notion of validity? The validity of a CG is at the central part of this article, so we want to know whether CG of real-world SLAs follow this notion of validity.

RQ3: Which difficulties appear when modelling SLAs defined in natural language? All real-world SLAs are expressed in natural language. Therefore, before checking their validity it is necessary to formalise them. With this question, we examine the problems that may appear in this step.

To answer these questions, we have modelled with iAgree and tried to validate up to 319 compensable guarantees that are described in natural language, in 24 different scenarios[8] belonging to three different domains, namely: cloud service providers (e.g., Amazon and Rackspace), non-cloud service providers (e.g., DHL and train companies), and B2B service outsourcing. As a complementary material we provide a workspace with the whole list of agreements modelled; in such a context, while in the Designer Studio tool a pre-loaded demonstration workspace is available with a subset of agreements, the whole dataset of modelled agreements[9] can be imported into the tool to be analysed. Next, we detail the results.

6.1 RQ1: Expressiveness of Compensable SLAs

Table 4[10] depicts a number of features regarding the 24 scenarios which have been studied. In a first block, we present (i) the number of guarantees that have the corresponding features, and (ii) the number of metrics which are involved in each guarantee, most usually one. In the second block, we present which metrics are involved in compensation functions (CF) for each scenario. Finally, in a third block, we present whether the scenarios do not hold validity criteria defined in Sections 3.2 and 4.2, which are discussed later in Section 6.2. Next we provide details for the kind of metrics included in the second block:

- The CF does only involve the SLO metric, which is the most usual case. As an example, if the SLO is $\text{availability} \geq 99\%$, then the penalty condition is $\text{availability} < 99\%$, and the proper penalty is $\text{Penalty} = 10\%$.
- The CF involves an aggregation of the SLO metric. As an example, in GNWT a penalty condition is defined as "*Less than 95 percent of severity-4 problems are resolved in 20 calendar days*".
- The CF involves the SLO metric multiplied by a constant value, which is another usual case. As an example, consider the expression $\text{Penalty} = 0.1 \times unavailablePeriod \times monthlyBill$.
- The CF involves other service and, possibly, different SLAs. As an example, in RENFE (the public Spanish train service) a penalty is expressed as "*If service is cancelled within 48 hours of scheduled time, RENFE is obliged to give an alternative transport, be either by train or by another mean of transport [...] or the devolution of the service cost.*". Note the reference to initiate another SLA (a contract for a new service) due to the alternative transportation.
- The CF involves value metrics from more than one billing cycle. As an example, in Movistar (an Spanish telecommunications company) a penalty is expressed as "*if the installation deadline is not fulfilled, the provider must give an automated compensation which consists of [...] a number of monthly fees equivalent to the number of months, or fraction, by which the deadline has been overcome*".

8. Documents used to model SLAs at https://goo.gl/yLvxo5
9. Available at https://goo.gl/PEvSTC
10. A detailed version of this table at https://goo.gl/8jXfn4

- The CF denotes whether the SLO is fulfilled during more than one billing cycle. As an example, in the Indian Baroda Bank, a reward condition is "*If Situation A is achieved for the consecutive 2 months following a situation E*", where such "situations" refer to different stages of SLO fulfilment.

Table 4 shows that our technique has three main limitations concerning the expressiveness of compensation functions. First, an SLO must be specified with just one service metric. Therefore, compensable guarantees marked with "2+" cannot be supported. Second, compensations have to be based on such SLO metric, so that its related utility precedence function can be properly evaluated. (ie. if CF is not based on the same SLO metric, it cannot be validated). Third, due to limitations in the CSP engine used in the analysis, metric domains are restricted to Integer type. This limitation can be overcome by transforming real domains to integers with a particular precision; e.g., a metric for a percentage that should be real, can be defined as an integer with domain 0..10000 assuming 10^{-2} of precision.

In addition, there are scenarios which can be supported by our technique, namely, the compensation expression may include other variables provided that they are independent from the SLO metric, since such variables do not have an impact on validity analysis (ie. the CF involves the SLO metric multiplied by a constant value as depicted in Table 4). As an example, let be an SLO specifying a maximum in the number of human resource substitutions (HRS) for a project, expressed as $\mathrm{HRS} \leq 2$. In this context, a compensation expression as $\mathrm{Penalty} = (\mathrm{HRS} - 2) \times \mathrm{HPR} \times 40$, where HPR stands for an hourly profile rate, can be supported because HPR is independent from HRS. In another example, note the independent MonthlyBill variable in the compensation expression $\mathrm{Penalty} = 0,1 \times \mathrm{unavailablePeriod} \times monthlyBill$. Both cases show that the independent variable can be even considered as a *constant* with regard to the SLO metric; as a matter of fact, they are omitted in the calculus for validation analysis with no effect in the result.

Despite these limitations, our tooling support can validate up to 242 compensable guarantees (76 percent of reviewed cases).

6.2 RQ2: Effectiveness of Proposed Validation Technique

After modelling the iAgree documents of the 24 real-world scenarios and using our proposed automated validation technique, we found that nine compensable guarantees were not properly defined in the original SLAs specified in natural language. Specifically, five were wrongly specified by Verizon, and four were wrongly specified by the outsourcing service hiring of the regional governments of: Northwest Territories of Canada, and Andalusia in Spain. These results are included in the third block of Table 4.

Among the 7 cases that are not consistent, in the Verizon scenario we find in three compensable guarantees (cf. `A14G13-G15` in the dataset) a wrong use of a "less than" symbol instead of a "less and equal" symbol in the definition of penalty conditions for the metric values intervals with higher penalty assignment. On the one hand, in `A14G13` the conditions that penalise with 10 and 25 service credits are $\mathrm{availability} \geq 95.000\%$ and $\mathrm{availability} < 94.999\%$, respectively. Thus, while an availability of either 95.000% or 94.998% would be penalised with 10 and 25 service credits, respectively, an availability of exactly 94.999% does not imply a penalty. Therefore, any of these penalty conditions should include 94.999 percent value and this is not the case. On the other hand, in `A14G16` and `A14G19` the problem is the lack of a lower limit value in the penalty conditions as follows: Penalty of 20 service credits if $\mathrm{availability} < 99\%$; Penalty of 40 service credits if $\mathrm{availability} < 98\%$, etc. Thus, the overlapping of penalty assignments makes it impossible to apply the right penalty for values of 97 percent or less.

In the GNWT scenario, two compensable guarantees (cf. `A01G04` and `A01G16` in the dataset) have the same kind of problem in the compensation condition by a wrong use of the expression "*20 percent less than*" instead of "*up to 20 percent less than*". The first one has been analysed as a running example to exemplify an invalid compensation function in Section 3.2. This wrong definition results in a reward defined just for a single metric value while other metric values with higher utility do not imply a reward.

Among the 2 cases which are not coherent, in the Sandetel outsourcing service scenario, two compensable guarantees (cf. `A16G01-G02` in the dataset) have the same kind of problem in the compensation definition because the penalized metric values are those that fulfill the SLO. Specifically, in `A16G01` the SLO is $\mathrm{Percent\ of\ Solved\ Interventions} > 90\%$ and the penalties apply for all the values for the metric that are greater than 90 percent and lesser than 95 percent. This results in a nonsense penalty for a hired company that is properly offering the service (i.e., while the SLO specified is being fulfilled, the service provider is penalised due to the wrong penalty conditions).

6.3 RQ3: Modelling Issues

During the recent years, we have found that the use of natural language in the reviewed SLAs of real-world scenarios[11] makes them susceptible to include semantic ambiguities. Among others, we have found the following problems that must be solved in order to obtain a formal iAgree model of the real-world scenarios:

6.3.1 Imprecise and Misleading Compensation Conditions and/or SLOs

As an example, OVH (cf. `A04` in the table and dataset) guarantees an availability of 99,999 percent but compensations only apply for 1-minute periods and starting after a 3-minute period of downtime. However, it is not clarified if the "3-minutes" exception might be considered for every single failure, or also in relation to the aggregated failure time. In our modelling (cf. `A04G01` in the dataset) we have considered that the 3-minutes initial exception is affecting to the aggregated failure time. Thus, to compute the compensation we use the accumulated unavailability, defined as the aggregated minutes of unavailability after the first three, divided by the total minutes in the month. In addition, the SLO is not precisely defined because considering a 31-days month, the 100 percent of availability in minutes are 44640 minutes, and the 99.999 percent of availability just permits 1

11. This study was comprised of 3 cases in [3], 5 in [23], and 24 in the current article. All of them are available at: https://goo.gl/yLvxo5.

minute of unavailability that will not be compensated due to the lack of an initial "3-minutes" exception. Therefore, as one compensation applies after 4 minutes of unavailability, the total minutes of availability without compensations is 44636 (i.e., a 99.991 percent of availability) and thus, the actual SLO would be $\text{availability} > 99.991\%$.

6.3.2 Usage of Imprecise Value

As in Rackspace and Host Europe (cf. `A03` and `A08` in the table and dataset, respectively), there are imprecise values. In these cases, the availability percentages are given as integer values or float values providing just one decimal. However, it is not clear whether they are simply rounding values for the sake of clarity, or not. For instance, Rackspace compensates with 5 percent of the service fees for each 30 minutes of Cloud Database Instance unavailability, after the first 0.1 percent of unavailability during the month (cf. `A03G02` in the dataset). Note that this 0.1 percent is equivalent to 43.2 minutes in months of 30 days and thus, it is not clear if the penalty applies after 43 or 44 minutes of unavailability.

6.3.3 Lack of SLOs

Many SLAs include compensable guarantees without an explicit SLOs, as the GNWT running example (cf. `A01` in the dataset). In these cases, a domain expert should infer the SLO from the compensation conditions: Fig. 8 includes the SLO $\text{EDP} < 120$ that we inferred from the penalty condition $\text{EDP} \geq 120$. Recently, Amazon has changed the AWS S3 SLA (cf. `A02` in dataset), and its current version does not state any SLO[12].

A side effect of representing in a machine-processable iAgree document the real-world SLAs defined in natural language is that some modifications must be done to face the three aforementioned modelling problems. In the following, we provide some modelling best practices that an expert should consider in the process of writing an iAgree document supported by our proposed technique.

As general rule, the expert should obtain a mathematical formula for both SLO and compensations. In case of SLOs specified as equations of the form $\text{UnavailableMinutes} == 0$ or $\text{availability} == 100$, as mentioned in Section 5.1, they should be transformed to an SLO of the form $\text{UnavailableMinutes} \leq 0$ or $\text{availability} \geq 100$, respectively, because this allows to infer a proper utility function for the metric.

This general rule must be applied wisely by making decisions to solve the modelling issues. These decision making could be as simple as in the following three scenarios: (i) in the GNWT scenario, the iAgree document of Fig. 8 is straightforwardly gathered from the GNWT-4 term of Fig. 2 by just considering the $\text{EDP} == 100\%$ as the scheduled delivery date; (ii) the aforementioned imprecise compensation function definition of CloudLock scenario can be easily solved by considering the metric $\text{monthly uptime percentage}$ as a kind of availability that is the SLO-related metric (cf. `A10` in the dataset); and (iii) the exception condition of OVH scenario can be solved by considering a new metric called $\text{Accumulated unavailability}$ as a kind of unavailability that does not include the first 3 minutes of unavailability (cf. `A04` in the dataset). Regarding the imprecise value definitions, we cannot model what is not expressed in the real-world SLAs. Therefore, we propose to consider that the specified values are rounded values despite of the impreciseness it implies in terms of compensations.

On the contrary, applying the general rule to other scenarios may lead to make more complex decisions. For instance, in GoGrid (cf. `A15` in the dataset), it is guaranteed the 100 percent of server uptime so that the costumers are compensated with a 100 percent of the failure time (in relation with the customer fee) in further service credits. As an example, a one hour failure of a virtual server, whose cost of 1GB RAM is \$0.08/GB Hour, will generate a credit of \$0.08 x 1 (GB) x 100 = \$8. In this case, two considerations are required: i) the customer fee should be omitted by our modelling because it is a constant value; and ii) server uptime and duration failure, in total hours, must be taken into account in each compensation term. This GoGrid scenario demonstrates that we must also include as best modelling practice the homogenization of metrics used in both: SLO and compensations conditions.

7 Related Work

In this article we have made the question "*How can compensations be automatically validated?*". As far as we know, other works have also approached this or a similar question, as commented in the following. A first point is how expressive are their approaches. Similar to ours, having a compensation including condition, penalty, and saturation, some works extend WS–Agreement as [9], [12], [19], others have a semantic approach as [10], [28], [29], [30], and most an abstract model [7], [11], [16], [22]. Other approaches only have one or another element, such as [5], [6], [13], [14], [15], [17], [18]. Pioneering this point, Leitner et al. in [4] introduce some features of penalty functions and describe up to four types of penalty functions. This work was the starting point of our initial formalisation [3] and modelling [23] approach, but including rewards and the properties to validate compensations. A second point is whether these approaches enable some kind of automated validation to avoid wrong specifications. One the one hand, Krotsiani et al in [9] extend WS–Agreement by means of penalty and renegotiation predicates, which are processed by means of a probabilistic timing automata. This allows to make predictive questions such as "*what is the probability to pay a penalty for the next week?*" on the event of incoming service requests. Thus, it is a different but complementary approach to ours. On the other hand, Kritikos et al. in [10] extend OWL with a semantic SLA model in which service levels are assigned both compensations and pricing. The validation, defined by means of SWRL derivation rules, allows to check (1) the monotonicity of compensations according to their metrics, and (2) the transitions between service levels mirror their pricing accordingly, so that lower levels are cheaper than higher ones. The overall objective is to minimise the costs on the event of transitions to adjust the quality-of-service. Third, regarding the best practices, we only have found the work of [29] which includes an extensive analysis on the question of legality on using WS–Agreement. Fourth and last point, it is the use of compensations in these works. Most of approaches try to avoid, one way or another, SLA violations and/or find

12. The service commitment of 99,9 percent in previous version (https://goo.gl/HGrGjH) has been removed in current version (https://goo.gl/KdB665)

adaptations to prevent them, in order to minimise the overall cost. Hussain et al. [13] try to minimise the number of violations, thus penalty costs. Ranaldo et al. [16] avoid violations while maintaining a competitive price. Labidi et al. [28] introduce CSLAONTO, a ontological SLA model defined by SWRL rules, which are processed by the Jess inference engine, to monitor and predict SLA violations, then adjusting service execution, in order to minimise the penalty risk. Serrano et al. [14] reduce the service price in case of request which may provoke a SLA violation. It also introduce the notion of a procedure for notifying penalties, including: (1) the actor in charge, (2) the method, and (3) the period for notification. Chard et al. [15] try to avoid malicious bidding strategies. Maarouf et al. [8] propose a formal model considering penalties of different metrics. Xiaoyong et al. [7] try to maximise the revenue by means of avoiding penalisations. Amokrane et al. [17] reduce costs in a green cloud environment. Narasayya et al. [18] try to minimise penalisations in multi-tenant relational databases-as-a-services. Garg et al. [11] try to maximise resource usage while the SLA is fulfilled, together with SLA enforcement, admission control and scheduling, and forecasting. Haq et al. [12] try to prevent SLA violations, by means of pro-active actions (adjusting infrastructure to avoid violations), and reactive actions (penalty enforcement). It introduces a agent-based framework based on a RuleML ontology which extends WS–Agreement. Rana et al. [29] identify how SLOs may be impacted by the choice of specific penalty clauses. Grabarnik et al. [6] propose a model intended to reduce costs of composite services, by means of considering a trade-off between penalty costs and fulfilment cost at a design-time choice of service suppliers. Paschke et al. [30] model an SLA which defines minimum and maximum thresholds to compensate SLAs unfulfilment or overfulfilment. Last, Buco et al. [5] propose an SLA management system that uses penalties to alert about potential cumulative penalty cost.

In business studies, utility function models are also analysed as they are strongly dependent on customer preferences and behaviour. Bar-Isaac et al. in [21] describe a business scenario with cost, customer expectations and reputation variables where reward function follows a non-monotonic behaviour (based on satisfying preferences from different customers). Similarly, Ren et al. analyse in [31] how utility function is obtained from customer objective function (i.e., customers timetable preferences affect how transactions distribute through commercial opening hours).

Finally, in authors' previous work [25], the validity of SLAs is formalised by considering a set of conflict-free guarantee terms that define valid assertions that must be fulfilled, i.e., an SLA is considered as valid if it is defined without conflicts between and within its SLOs. Examples of conflicts are: *Resolution Hours* $<= 4h$ *AND* > 4; or *MUP* $<$ *99.95 percent IMPLIES MUP* $\geq$ *99.95 percent*; however, such validity does not consider compensations.

8 Conclusions and Future Work

Using the presented formalisation and technique, the assessment conducted over 24 real-world SLAs allowed us to identify 9 situations that lead to wrong compensable SLAs and whose mistakes could be automatically identified. The proposed validity criteria can be used for current approaches of other authors as well as the underlying CSP-based technique that enables the automation. Thus, activities that rely on compensations such as adjust the billing or predict SLA violations can be carried from now on in a safer and more reliable way. Regarding the expressiveness limitations of our proposal, the existence of some kind of terms unable to be specified with our abstract model encourage us to expand both the abstract model and iAgree. Finally, we would like to highlight that checking the validity is just one step to debug real-world SLAs; as next steps, to allow temporal-aware compensations considering [32], and to diagnose and suggest solutions for the problems found will be addressed with a similar approach than [26] where we automate the diagnosis of SLA violations.

Acknowledgments

We would like to thank the reviewers for their comments that have significantly improved the quality of the article. Partially supported by grants TIN2015-70560-R (MINECO/FEDER, UE), TIN2016-81978-REDT (MINECO), P12-TIC-1867 (Andalusian R&D&I).

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) (v. GFD-R.192)," Open Grid Forum (OGF), https://www.ogf.org/documents/GFD.192.pdf, Oct. 2011.

[2] J. García-Galán, P. Trinidad, O. F. Rana, and A. Ruiz-Cortés, "Automated configuration support for infrastructure migration to the cloud," *Future Generation Comput. Syst.*, vol. 55, pp. 200–212, Feb. 2016.

[3] C. Müller, A. M. Gutiérrez, O. Martín-Díaz, M. Resinas, P. Fernández, and A. Ruiz-Cortés, "Towards a formal specification of SLAs with compensations," in *Proc. OTM Confederated Int. Conf., "On the Move Meaningful Internet Syst."*, 2014, pp. 295–312.

[4] P. Leitner, W. Hummer, and S. Dustdar, "Cost-based optimization of service compositions," *IEEE Trans. Serv. Comput.*, vol. 6, no. 2, pp. 239–251, Apr. 2013.

[5] M. J. Buco, R. N. Chang, L. Z. Luan, C. Ward, J. L. Wolf, and P. S. Yu, "Utility computing SLA management based upon business objectives," *IBM Syst. J.*, vol. 43, no. 1, 2004.

[6] G. Grabarnik, H. Ludwig, and L. Shwartz, "Management of service process QoS in a service provider - service supplier environment," in *Proc. IEEE Int. Conf. E-Commerce Tech.*, Jul. 2007, pp. 543–550.

[7] Y. Xiaoyong, T. Hongyan, L. Ying, J. Tong, L. Tiancheng, and W. Zhonghai, "A competitive penalty model for availability based cloud SLA," in *Proc. 8th IEEE Int. Conf. Cloud Comput.*, 2015, pp. 964–970.

[8] A. Maarouf, B. El qacimy, A. Marzouk, and A. Haqiq, "A novel penalty model for managing and applying penalties in cloud computing," in *Proc. 12th IEEE/ACS Int. Conf. Comput. Syst. Appl.*, 2015, pp. 1–6.

[9] M. Krotsiani, C. Kloukinas, and G. Spanoudakis, "Validation of service level agreements using probabilistic model checking," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2017, pp. 148–155.

[10] K. Kritikos, D. Plexousakis, and P. Plebani, "Semantic SLAs for services with Q-SLA," in *Proc. Comp. Sci. 2nd Conf. Cloud Forward: From Distrib. Complete Comput.*, 2016, vol. 97, pp. 24–33.

[11] S. Garg, S. K. Gopalaiyengar, and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.*, 2011, pp. 371–384.

[12] I. U. Haq, E. Schikuta, I. Brandic, A. Paschke, and H. Boley, "SLA validation of service value chains," in *Proc. Int. Conf. Grid Cloud Comput.*, 2010, pp. 308–313.

[13] W. Hussain, F. K. Hussain, O. Hussain, R. Bagia, and E. Chang, "Risk-based framework for SLA violation abatement from the cloud service provider's perspective," *Comput. J.*, vol. 61, pp. 1306–1322, 2018.

[14] D. Serrano, S. Bouchenak, Y. Kouki, F. A. de Oliveira Jr, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, and P. Sens, "SLA guarantees for cloud services," *Future Generations Comput. Syst.*, Jan. 2016, pp. 233–246.
[15] K. Chard and K. Bubendorfer, "Co-operative resource allocation: Building an open cloud market using shared infrastructure," *IEEE Trans. Cloud Comput.*, Jul. 2016, p. 1.
[16] N. Ranaldo and E. Zimeo, "Capacity-driven utility model for service level agreement negotiation of cloud services," *Future Generations Comput. Syst.*, vol. 55, pp. 186–199, 2016.
[17] A. Amokrane, et al., "Greenslater: On satisfying green SLAs in distributed clouds," *IEEE Trans. Netw. Service Manage.*, vol. 12. no. 3. pp. 363–376, Sep. 2015.
[18] V. Narasayya, et al., "Sharing buffer pool memory in multi-tenant relational database-as-a-service," *Proc. VLDB Endowment*, vol. 8, no. 7, pp. 726–737, Feb. 2015.
[19] A. García and I. Blanquer, "Cloud services representation using SLA composition," *J. Grid Comput.*, vol. 13, pp. 35–51, Mar. 2015.
[20] M. Cho, et al., "A new framework for defining realistic SLAs: An evidence-based approach," in *Proc. Int. Conf. Business Process Manage.*, 2017, pp. 19–35.
[21] H. Bar-Isaac and J. Deb, "What is a good reputation? career concerns with heterogeneous audiences," *J. Ind. Org.*, vol. 34. pp. 44–50, 2014.
[22] P. Leitner, et al., "Cost-efficient and application SLA-aware client side request scheduling in an infrastructure-as-a-service cloud," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 213–220.
[23] C. Müller, P. Fernandez, O. Martin Diaz, A. M. Gutierrez, M. Resinas, and A. Ruiz Cortes, "Specifying compensations with WS-agreement," *IEEE Latin America Trans.*, vol. 15, no. 7, pp. 1335–1341, Jun. 2017.
[24] A. H. M. Ter Hofstede, et al., "How to formalize It? formalization principles for information system development methods," *Inf. Softw. Technol.*, vol. 40, pp. 519–540, 1998.
[25] C. Müller, et al., "Comprehensive explanation of SLA violations at runtime," *IEEE Trans. Services Comput.*, vol. 7, no. 2, pp. 168–183, Apr.-Jun. 2014.
[26] C. Müller, et al., "Automated analysis of conflicts in WS–agreement," *IEEE Trans. Services Comput.*, vol. 7, no. 4, pp. 530–544, Oct.-Dec. 2014.
[27] N. Nethercote, et al., "MiniZinc: Towards a standard CP modelling language," in *Proc. 13th Int. Conf. Principles Practice Constraint Programm.*, 2007, pp. 529–543.
[28] T. Labidi, et al., "CSLAOnto: A comprehensive ontological SLA model in cloud computing," *J. Data Semantics*, vol. 5, pp. 179–193, 2016.
[29] O. Rana, et al., "Managing violations in service level agreements," in *Grid Middleware Services: Challenges Solutions*, Berlin, Germany: Springer, 2008, pp. 349–358.
[30] A. Paschke and M. Bichler, "Knowledge representation concepts for automated SLA management," *Decision Support Syst.*, vol. 46, pp. 187–205, Dec. 2008.
[31] F. Ren and M. Zhang, "Bilateral single–issue negotiation model considering nonlinear utility and time constraint," in *Decision Support Systems*, vol. 60, Amsterdam, The Netherlands: Elsevier, Apr. 2014.
[32] O. Martín-Díaz, et al., "An approach to temporal-aware procurement of web services," in *Proc. Int. Conf. Service-Oriented Comput.*, 2005, pp. 170–184.

Carlos Müller received the PhD degree in computer science from the University of Sevilla, Spain. He is a lecturer and member of the Applied Software Engineering Group (ISA, www.isa.us.es) at University of Sevilla, Spain. His current research line includes the automated analysis of service level agreements and the application of such analysis at SLA design and monitoring.

Antonio M. Gutierrez received the PhD degree in computer science from the University of Sevilla, Spain. He is a member of the Applied Software Engineering Group (ISA, www.isa.us.es) at University of Sevilla, Spain. He worked as engineer for several companies before joining the academia. His research interests are related to service oriented computing and business process management.

Pablo Fernández received the PhD degree in computer science from the University of Sevilla, Spain. He is a lecturer and member of the Applied Software Engineering Group (ISA, www.isa.us.es) at University of Sevilla, Spain. His current research is focused on the automated governance of organizations based on service level agreements and commitments.

Octavio Martín-Díaz received the PhD degree in computer science from the University of Sevilla, Spain. He is a lecturer and member of the Applied Software Engineering Group (ISA, www.isa.us.es) at University of Sevilla, Spain. His current research line includes purchasing in cloud computing and service oriented computing, specifically aspects related to time management in service level agreements.

Manuel Resinas received the PhD degree in computer science from the University of Sevilla, Spain. He is a lecturer at the University of Sevilla, Spain. His current research lines include analysis and management of service level agreements, business process compliance, and process performance management. Previously, he worked on automated negotiation of service level agreements.

Antonio Ruiz-Cortés is a full professor of software and service engineering and he heads the Applied Software Engineering Group, University of Sevilla. His current research focuses on service-oriented computing, business process management, testing and software product lines, being the recipient of the Most Influential Paper of SPLC 2017 award. He is an associate editor of Springer Computing. Contact him at aruiz@us.es.