# SLA MANAGEMENT OF NON-COMPUTATIONAL SERVICES

## DIGITAL TRANSFORMATION OF SERVICES DRIVEN BY SLAS

Thesis written by

ANTONIO MANUEL GUTIÉRREZ FERNÁNDEZ

and supervised by

Dr. Antonio Ruiz Cortés and Dr. Manuel Resinas Arias de Reyna

Universidad de Sevilla

Dpto. de Lenguajes y Sistemas Informáticos

E.T.S.I. Informática

May 2018

**Support:** Pre-doctoral scholarship, and scholarships for research visits granted by the Spanish Government under CICYT project TAPAS (TIN2012–32273). Additional support for research visits and attending conferences provided by: the European Commission project: RISEBPM (H2020–645751); the Spanish Government project BELI (TIN2015–70560–R); and the Andalusian Government project COPAS (P12–TIC–1867).

*A toda mi familia por su apoyo en todo momento*

# Agradecimientos

Según Bernard Chartres, cualquier avance científico "no es más que mirar desde los hombros de unos gigantes". No está claro si habré conseguido mirar más lejos, pero tengo claro que mi aportación nunca es más que una pequeñisima luz imperceptible al calor de la suma de fuegos con la que mi familia, amigos y colegas me ha alumbrado siempre.

El camino formal empezó de la mano de Manolo, sin el que me hubiera sido imposible llevar a buen término esta tarea. Con su dirección he aprendido a ordenar, retorcer y exprimir las ideas pero también tengo que agradecerle la paciencia para soportar mi necesidad de reafirmar mis propias ideas a modo de espejo. Pero esta tesis empezó a formarse hace más de 10 años. Hablar de SLAs, preferencias de calidad y despliegue de servicios delante de una tostada de jamón, aceite y tomate con Pablo era, claro, el principio de algo. Mucho después, no puedo evitar pensar que en cierta forma, el corazón y la razón de esta tesis son hijos de ambos (aunque putativo si se me permite el decirlo). También ha contado con las visitas ocasionales del tito José Antonio, aportando siempre sensatez pero combinada con ideas y humor. Y en esta familia no falta tampoco, y no lo digo por la edad, el .$^{a}$buelo.$^{A}$ntonio, poniendo orden y dirigiendo desde la tramoya los hilos e ideas de ésta criatura.

En un camino tan largo, por supuesto, han sido muchos también los primos (sin segundas) que han aportado su luz. Se me hace difícil dar un nombre primero, pero han sido muchos cafés que Alejandro ha soportado con mis incesantes inquietudes. Han sido también muchas las líneas de Latex con Carlos, que tanto (o tan poco :D) me han hecho aprender a formatear. Y David, aportando orden, no sólo a las ideas científicas, sino personales. Y muchas han sido también las horas juntos en el aula ISA, tanto con Cristina, iluminándome con su férrea disciplina y capacidad de trabajo, como Adela con su alegría y comprensión. Y antes, en el aula SUN, los buenos ratos profesionales con Ana Belén, José Galindo y Jesús Galán. Y como, además de hablar, también me gusta escuchar, he tenido la suerte de aprender de Joaquín y Sergio, dos grandes 'doers'. Y todo, sin olvidarme de los buenos momentos que he podido pasar con todos los compañeros del grupo, Octavio, Amador, José Mari, Pablo T, Bea, Margarita y, un poquito más recientemente, Bedilia, Javi, Alfonso y Antonio G. En realidad son muchos más nombres, incluyendo a los compañeros de LSI, como para nombraros a todos, gracias.

Esta tesis tampoco hubiera sido la misma sin las estancias que he disfrutado en Essen, Vienna y Rio de Janeiro. Gracias a Clarissa y Andreas, Scharahm y Hong Ling y Flavia y Claudia por brindarme la oportunidad de formarme con vosotros y espero

poder seguir extendiendo las colaboraciones que hemos comenzado.

Y por último, quiero acordarme de todos los que no saben ni de que va ésta tesis pero han sido igualmente importantes. Mi familia y mis amigos. Por un lado, mis padres, Amparo y José, y mi hermano Pepe que siempre han confiado en mi y puesto todo de su parte para que llegue tan lejos como proponga. Pero también mis amigos que no han dejado de preguntarme para cuando terminaba, Juanje, Pablo, Silvia, Alejandro y Jesús. Por último quiero nombrar a la persona que más ha soportado este camino, Ana. De todo corazón, gracias.

Guti

# Acknowledgments

Bernard Chartres said that any scientific is "looking farther on the shoulders of giants". I am not sure if I succeed to look farther but I am sure that my addition is just an indiscernible light in the sum of fires with which my family, friends and colleagues have always lighted me.

The formal path of this thesis started together with Manolo and I could have not finished it without his driving. I have learned to sort, twist and squeeze ideas but I have also to thank him the patience to hear my need of reaffirm my ideas. But this thesis started also to take shape more than 10 years ago. Chat in the breakfast about SLAs, quality preferences and services deployment with Pablo and a piece of bread with olive oil, tomato and jamon was the beginning of something. A long time after, I cannot help thinking that the heart and mind in this dissertation are children of both (although putative if i may say). Of course, there have also been the occasional visits of the uncle, José Antonio, which has always provided sanity together with a piece of ideas and good mood. And to complete the family, there is also a grandfather (not because his age!), Antonio, who has put order in the family and has managed from the stage the ideas and plot of this dissertation.

In a so long path, there have been, of course, many cousins providing their light. It is not easy to start with a name, but there have been a lot of coffees with Alejandro to hear my often questions. There have been also a lot of Latex lines with Carlos, with who I have learned so much (or so few) to format. And a lot of Napolitanas with David, providing support to my ideas not just scientific but personals. And also long time in the ISA office with Cristina, providing the light of iron discipline and, Adela, always with joy and empathy. And sometime before, in the SUN office, the "technical" fun with Ana Belén, José Galindo and Jesús Galán. I not only like talk but also hear so I have also been lucky to hear from the 'doers' Joaquín and Sergio. And, of course, I do not want to forget any of my ISA colleagues, Octavio, Amador, José Mari, Pablo T, Bea, Margarita and, a bit more recent, Bedilia, Javi, Alfonso and Antonio G. There are really much more names to say, including my colleagues from the LSI department. Thank you.

This dissertation would have neither been the same without my research stays in Essen, Vienna and Rio de Janeiro. Thank you to Clarissa and Andreas, Schahram and Hong Ling and Flavia and Claudia for giving the opportunity to learn with you and I hope to collaborate further with you.

And last, I want to remind all the people that dont́ know about what this dissertation discuss but are equally important for the result. My family and friends n the one side, thank you to my parents, Amparo y José, and my brother, Pepe, because you have always trusted on me and supported to handle any achieve I propose. And also, thank you to all my friends that never stopped to asked when I was going to finish, Juanje, Pablo, Silvia, Alejandro y Jesús. And lastly, I want to name the person who longer suffered this path, Ana. My sincere thanks.

Guti

# Resumen

El incremento en el uso de arquitecturas orientadas a servicios en los últimos 15 años ha propiciado la propuesta de numerosas técnicas para automatizar y dar soporte al uso de dichos servicios. Un elemento fundamental en la provisión de servicios es el Acuerdo de Nivel de Servicio (ANS), donde se formalizan los requisitos y garantías de consumidor y proveedor respecto del rendimiento del servicio. Las propuestas para servicios computacionales, además de proveer modelos formales para describirlos, proponen la automatización de las diferentes etapas del ciclo de vida del ANS, tales como la negociación de las garantías para crear un ANS, el despliegue de servicios basados en el ANS, o la gestión de los recursos para cumplir las garantías provistas en el mismo.

Sin embargo, en los servicios tradicionales, no computacionales, es decir, los servicios que no son ejecutados por recursos computacionales, tales como los servicios de logística o de desarrollo de software, la gestión de sus ANSs todavía se realiza por medios ad-hoc. Así, las soluciones existentes no pueden ser reutilizadas por diferentes servicios. Y, en la mayoría de los casos, esta gestión se hace de manera manual (p.e. revisión de los objetivos acordados en los ANSs de servicios de transporte), por lo que la evaluación de estos ANSs es susceptible a errores y se suele retrasar respecto a la ejecución del servicio (p.e. cuando el ANS ha finalizado), por lo que no se pueden tomar acciones preventivas para evitar el incumplimiento del ANS o estas acciones no son rentables. En estos escenarios, aparecen, además, acuerdos marco para un periodo largo (p.e. 1 aõ), durante el cual pueden aparecen ANSs relacionados con éste para un periodo más específico y el análisis de la coherencia entre acuerdos marco y acuerdos específicos es complicada de hacer durante la ejecución del servicio.

En esta tesis, nos proponemos automatizar parcialmente la gestión de los ANSs de servicios no computacionales.

Así, por un lado, proponemos que los modelos para servicios computacionales se extiendan a servicios no computacionales, de manera que permitan describir la operativa del servicio y sus garantías. Y, por otro lado, basado en estos modelos, proporcionamos el diseño de operaciones para gestionar el ciclo de vida de los ANS. Concretamente, estas operaciones se basan en las fases de despligue y evaluación del ANS.

De forma específica, esta tesis propone tres contribuciones principales. Primero, (A) extender iAgree para dar soporte al modelado de los ANS de servicios no computacionales. Segundo, (B) dar soporte al ciclo de vida de dichos ANS mediante la formalización de las operaciones citadas (configuración del servicio basada en el ANS y

monitorización del mismo) y, a partir de estas operaciones, implementamos una arquitectura de referencia para estas operaciones. Y, por último, (C) proveemos el modelado de la relación entre acuerdos marco y específicos que relacione sus términos junto con la formalización de las operaciones para el análisis que aparecen entre ellos.

Otros aspectos del ciclo de vida del servicio y del ANS, como la gestión de los recursos para mejorar el rendimiento del servicio o el uso de técnicas (como machine learning) para la predicción del cumplimiento de los ANSs están fuera del contexto de esta tesis, pero se plantean como futuras líneas de extensión.

Este trabajo se ha basado en ANSs reales de diferentes dominios, tales como servicios de Transporte y Logística, proveedores de Cloud or outsourcing de desarrollo TIC, que se han utilizado para validar las propuestas. Además, las contribuciones presentadas se han aplicado en el contexto de proyectos reales de soporte de sistemas TIC.

# Abstract

The rise of computational services in the last 15 years brought the proposal of a number of techniques to automate and support their enactment. One key element in services is the Service Level Agreement (SLA), where the requirements of service customer are matched with the performance levels from the service provider to define service level guarantees and related responsibilities. The proposals from computational domains are oriented to automate the different stages in the SLA Lifecycle, such as the negotiation of terms which will form the SLA, the deployment of services based on the SLA artifact or the management of computational resources to accomplish SLA goals on runtime.

However, traditional non-computational services, that is, services which are not performed by computational resources, such as logistics or software development services, are still supported by ad-hoc mechanisms. Therefore, the existing solutions for the management of their SLAs cannot be reused for other services. This management is usually manually performed (e.g.: reviewing of the goals of an SLA in transport service), so their evaluation is error-prone and delayed regarding the service execution (e.g.: when the SLA is finished), so preemptive actions to avoid SLA violations cannot be taken or/and are expensive to perform. Furthermore, these SLAs are sometimes described on a long term basis (frame agreements), and related SLAs can appear for a shorter term (specific agreements) and the analysis of the validity among them is complex to perform on runtime.

In this dissertation, we aim at partially automate the management of SLAs in non-computational services.

On the one hand, we suggest that existing models for computational services can be extended to non computational services and enable the description of the service operative and their guarantees. And, on the other hand, we provide a design for operations to partially support the SLA Lifecycle, based on the previous models. Specifically, these operations are mainly focused on the deployment and fulfillment stages of the SLA.

Therefore, the contributions of this dissertation are three. First, (A) providing a model to describe Service Level Agreements of non computational services, as an extension of iAgree, an existing model for SLAs of computational services. Second side, (B) supporting the SLA Lifecycle with the design of the aforementioned operations (service configuration based on SLA and monitoring of SLA) and implementing a reference architecture for such operations. And, lastly, (C) providing a model for frame and specific agreements which relates their terms and formalises the analysis opera-

tions among them.

Other related operations of the service lifecycle as the management of resources to improve service performance or the use of novel techniques (such as machine learning) to predict the SLA accomplishment are out of the scope of this thesis but planned as future line of extension.

The current dissertation has been based on real SLAs from different domains, such as Transport & Logistics, public Cloud providers or IT Maintenance outsourcing, which have been used to validate the proposal. And, furthermore, the contributions have been applied in the context of real IT Maintenance outsourcing projects.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **API** | Application Programming Interface. |
| **BAM** | Business Activity Monitoring. |
| **BI** | Business Intelligence. |
| **BP** | Business Process. |
| **BPEL** | Business Process Execution Language. |
| **BPM** | Business Process Management. |
| **BPMN** | Business Process Modelling Notation. |
| **BPMS** | Business Process Management System. |
| **CRM** | Customer Relationship Management. |
| **CSP** | Constraint Satisfaction Problem. |
| **EPC** | Event-driven Process Chain. |
| **ERP** | Enterprise Resource Planning. |
| **GT** | Guarantee Term. |
| **IT** | Information Technology. |
| **KPI** | Key Performance Indicator. |
| **PAIS** | Process Aware Information System. |
| **PPI** | Process Performance Indicator. |
| **SCU** | Social Computing Unit. |
| **SLA** | Service Level Agreement. |
| **SLO** | Service Level Objective. |
| **SOA** | Servie Oriented Architecture. |
| **UML** | Unified Modeling Language. |

# Part I

# Preface

# Introduction

*Victory belongs to the most persevering.*

*Napoleon Bonaparte (1769-1821),*

## 1.1 Research Context

In an intuitive manner, a service is the bundling or packeting of something that is consumed by someone, who requests and benefits from it and there is a provider which take responsibility in delivering it according to some kind of agreement. In the last decades, the use of computational services over Internet has enabled to make software consumers independent from providers, specially on the global Internet. Together with the rise of computational services, the definition of models and protocols to automate the management of Service Level Agreements (SLAs) has been a research topic during the last 15 years [6, 122]. SLAs are a fundamental key on the service interaction, as they match the requirements of service customer and the performance levels from the service provider to define service level guarantees and related responsibilities. The proposals from computational domains are oriented to automate the different stages in the SLA Lifecycle.

The SLA Lifecycle starts with the description of customer or provider performance preferences which results in the definition of a formal agreement, sometimes after a negotiation process among these parties. Then, the service is enacted according to the SLA and its fulfillment has to be evaluated during its validity period. Different mechanisms have been proposed to automatically negotiate guarantee terms based on domain information or the management of resources to accomplish SLA goals on runtime. This thesis has been developed in the context of the research group Applied Software Engineering (Ingeniería del Software Aplicada-ISA) of the Universidad de Sevilla, and it proceed from the research line within the area of analysis of computational services area, pioneered by A. Ruiz [156], O. Martín [116], Pablo Fernandez [60], Manuel Resinas [153] and Carlos Muller [122].

However, traditional non-computational services, that is, services which are not performed by computational resources, such as logistics or software development services, are still supported by ad-hoc mechanisms. Therefore, the existing solutions for the management of their SLAs cannot be reused for other services. In this dissertation we are going to base on non-computational services which are based on the provisioning of known business processes. These services can be intra-organisationally provided or partial or fully outsourced (i.e. business process outsourcing, BPO). Like computational services, their execution is regulated by SLAs and supported by specific software [81, 115]. In this case, since non-computational services are process-oriented, the software that supports them is usually a process-aware information system (PAIS) such as ERPs, Service Desk Management Systems, CRMs, or business process management systems (BPMSs). However, unlike computational services, this management is usually manually performed (e.g.: reviewing of the goals of an SLA in transport service), so their evaluation is error-prone and delayed regarding the service execution (e.g.: when the SLA is finished), so preemptive actions to avoid SLA violations cannot be taken or/and are expensive to perform. Furthermore, these SLAs are sometimes described on a long term basis, as a frame agreement for the service, and related SLAs can appear for a shorter term and the analysis of the validity between SLAs in different time terms is complex to perform on runtime.

One example of non-computational service is the transport and logistic of goods, where people take action to deliver goods from one point to another together mechanical resources such as cars for driving along a city or conveyor belts for movement inside a factory facility. The execution of manual tasks hinders the automation of their lifecycle, including the monitoring of their performance or the violation of their SLA. These difficulties are decreasing as with the digital transformation, the information of processes and services is increasingly supported by information systems.

A PAIS with SLA-aware capabilities, i.e. an SLA-aware PAIS, is a PAIS that uses explicit definitions of SLAs to enable or improve the automation of certain tasks related to both the SLAs and their fulfillment such as performance monitoring, human resource assignment or process configuration [179]. For instance, an SLA-aware PAIS could be automatically instrumented according to the metrics defined in the SLA, so that when there is a risk of not meeting an SLO, an alert is raised allowing the human actors involved in the process to take measures to mitigate the risk. Another example could be the automated configuration of the process, e.g. removing or adding activities, executed by the SLA-aware PAIS depending on the conditions of the SLA agreed with the client. Apart from the benefits derived from the automation of these tasks, the need for a SLA-aware PAIS becomes more critical in a business-process-as-a-service scenario. A business-process-as-a-service (BPaaS) is a new category of cloud-delivered service, which, according to Gartner [73], can be defined as "the delivery of BP services that are sourced from the cloud and constructed for multitenancy. Services are often automated, and where human process actors are required, there is no overtly dedicated labour pool per client. The pricing models are consumption-based or subscription-based commercial terms. As a cloud service, the business-process-as-a-service model is accessed via Internet-based technologies." In this setting, the conditions of the SLA agreed with each client may vary. Therefore, it is crucial for the PAIS that supports the business-process-as-a-service to behave according to the SLA agreed with the client. An example could be the prioritisation of the execution of tasks for those clients whose SLAs have bigger penalties if they are not met.

After the analysis of more than 20 SLAs in different scenarios and projects, such as IT Maintenance, Transport & Logistics or BPaaS solutions research, different problems have been identified which lay the ground for this dissertation. On the one side, as several tasks in the service execution involved human interaction, the related SLA documents, including the service context, metrics, SLOs and compensations, are described with natural language. This hinders an automated analysis and understanding of these documents. On the other side, non computational services include the use of heterogeneous resources, such as different information systems, human resources or devices.

## 1.2 Research Problems

The main goal for this dissertation can be stated as follows.

**Dissertation Goal**

*To partially automate the management of SLAs in non-computational services.*

In order to achieve this goal, we have identified the following two challenges to be addressed in this dissertation:

- Provide a model to support the automation of the SLA management for non computational services. This model has to enable the description fo business processes together with metrics related to them in order to define the guarantees provided for involved parties. Furthermore, the relationship between SLAs for different time periods (long and short term) has to be addresses in the model.

- Address the management of SLA Lifecycle for non-computational services, providing systems to support it during the phases where the SLA is considered active: deployment and fulfillment. These phases include the operations related to the enactment of the service together with the evaluation of the included guarantees and the decision making to accomplish them.

Our approach to address the aforementioned challenges is highly inspired in the solutions proposed to problems of automated management of computational services. As outlined in the above discussion, in the automation of the Service Level Agreement Management for non-computational services, a number of problems needs to be considered. Regarding the management of SLA Lifecycle, we focus on the late phases of the SLA lifecycle, that is, deployment and fulfillment of SLA as the SLA creation phases have been deeply analysed by Resinas [153] and Muller [122]. Regarding these stages, we analyse the impact of a more **specific agreement** within a **frame agreement** context, the **adaptation** of service according to the SLA and the **monitoring** the accomplishment of SLA during the *fulfillment* phase. The optimisation of resources to ensure the accomplishent of the SLOs is out of the scope of this dissertation. Therefore, in order to address the identified challenges, we identify four research questions related to the real world scenarios. The current dissertation will address the following four research questions:

**Research question 1.** How can we model a SLA for different domains supporting its automatic management?

**Research question 2.** How can we model the relationships between frame and specific SLAs together with the relationships that appear between their terms and goals?

**Research question 3.** How can we automatically configure and adapt the underlying information systems based on the SLA for non-computational services?

**Research question 4.** How can we automate the monitoring of SLAs and their guarantees for non-computational services?

These research questions are analysed and answered in the following chapters.

In the Table §1.1, we summarize the relationships between the challenges identified in existing scenarios, the research questions addressed and the contributions proposed in this dissertation which are explained in next Section.

Regarding the modelling of SLAs, we identify two different research questions. First, the **Research Question 1** aims to model non-computational services including service workflow, the related service metrics and the guarantees over them. While, the **Research Question 2** goals modelling different levels of SLAs that appear naturally on frame agreements for non-computational services but also identify related analysis operations between SLAs in different levels.

Regarding the operations to support the SLA Lifecycle, we have identified two different important mechanisms to support the deployment of the SLA and its fulfillment. First, in **Research Question 3**, we aim to provide mechanisms to deploy service driven by the SLA terms, such as adapting service workflow or resource assignment to party requirements. And, in **Research Question 4**, we goal to provide a mechanism to evaluate the fulfillment of the SLA, based on the monitoring of its SLOs.

| **Goal** | **Digital Transformation of Services driven by SLAs** | | | |
|---|---|---|---|---|
| **Challenges** | Modelling to automate the management of SLAs | | Automated Management of Services | |
| **Research Questions** | Q1: SLA Models for Business Processes | Q2: SLA Multilevel | Q3: Monitoring SLA accomplishment | Q4: Configuring service based on the SLA |
| **Contributions** | *Chapter 4: SLA Models for BPs* | *Chapter 6: Frame Agreements* | *Chapter 5: Business Processes driven by SLAs* | |

Table 1.1: Overview of the work performed in this dissertation

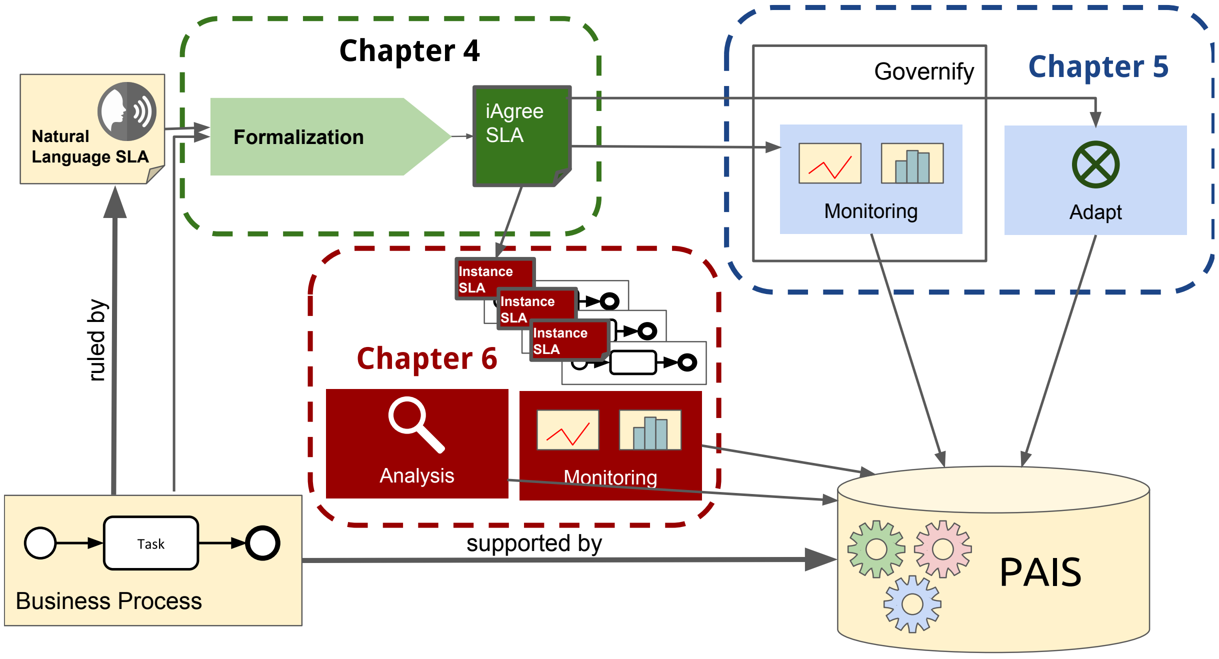


Figure 1.1: Contributions diagram

## 1.3 Contributions

Our approach to the aforementioned issues is highly inspired in the solutions proposed to problems of automated management of computational services. In order to achieve such a goal we have used different real scenarios to design artifacts to support this goal. Specifically this dissertation provides the contributions depicted in Figure §1.1.

First, in order to address the Research Question 1, an SLA model based on iAgree is proposed to support non-computational scenarios, which provides the service description, metrics and SLOs definitions for different parties and facilities to enable their analysis independent on the supporting information systems. The description of this model and how formalise a natural language SLA into this model is described in chapter §4.

Second, we also describe the operations to configure and monitor the service execution regarding its related SLA, together with providing a reference architecture to support them. This architecture consists of: (1) a monitoring component, which allows the monitoring of service performance independent of the supporting system; (2) a service adapter which allows the enactment of different service instances based on different SLAs of the same service. The described monitoring component addresses the Research Question 3, while the adapter addresses the Research Question 4, and both are described in the chapter §5 .

And third, we provide a model extension to describe frame and specific SLAs. Together with this model, we formalise the operations related to manage the lifecycle of more specific SLAs related to a frame SLA in the chapter §6, such as the conformance between a frame and a specific SLA to address the Research Question 2.

The introduced contributions have been developed as component architectures. These prototypes have been validated with a number of real SLAs (over 20) from different domains and further applied to real IT Maintenance outsourcing projects.

## 1.4 Research Methodology

This dissertation is conducted in the context of the Information Systems research area. The research in this discipline commonly use design science methodology for the proposal of innovative architectures solutions to stated problems. Design science appear fundamentally in engineering to address problems that are considered complex because they do not have an strict formulation and their solution only can be evaluated by utility. Therefore, it seeks create innovations to manage effectively and efficiently information systems. Some general guidelines for this methodology are introduced in [89]. In summary, this methodology proposes a sequence of expert activities to analyse existing problems and create an innovative product. Then, the product enables to re-evaluate the problem with new opportunities to improve the design in an iterative process. The phases suggested by Peffer et al. [146] in this methodology are the following:

1. **Identification of relevant problems.** The relevance of the research is based on the awareness of a problem that is of value for a community. The interest may come from industry or a research discipline. The general relevance of the research presented in this dissertation comes from the interest of different industrial partners that deal with the limitations on the state-of-the-art solutions to manage the lifecycle of SLAs, regarding specially the evaluation of SLA accomplishment of non-computational services identified in Chapter §5.

2. **Proposal of research goals.** After the analysis of certain scenarios, in this phase, we

evaluate related works and propose a solution to the stated problem which is used as a starting point for the research efforts. Abstract designs can be proposed in this phase. In this dissertation, we propose the research goals identified in previous Section §1.2, together with the Figure §1.1 to identify the expected contributions.

3. **Artifact design and development.** The goal of this phase is to produce a viable artifact including models, operations and software products following the previous proposal. This artifact can take the form of models, methodologies or software architectures. In our dissertation, we propose a model to describe SLAs together with the description of monitoring and configuration operations and an architecture to implement them (Chapters §4 and §5.

4. **Demonstration.** Once the artifacts have been developed, they are applied to the analysed scenarios to demonstrate that they can conduct the stated problems. We apply our model, which extends iAgree, to the IT Maintenance outsourcing in the Section §4.4 and implements the proposed operations in Chapter §5 to demonstrate the compliance of our models with the identified requirements.

5. **Evaluation.** Once demonstrated the feasibility of the artifacts, they are evaluated regarding further scenarios to validate the goals proposed in phase 2. Then, deviations from expectations are gathered and the additional information gained in the construction and running of the artifact is used to another iteration. In the Section §4.5, we describe more than 10 different SLAs from different scenarios, apart from the example one, to verify that our model can be applied to non-computational scenarios independently of the domain. This evaluation has been extended in the context of a transfer of technology project with local government and in an European project with logistics partners.

6. **Communication of research.** As the artifacts are proved effective, it is expected that they are communicated in relevant fora for the topic. In the case of this dissertation, we have published different contributions in relevant conferences and journals. Specifically, our proposal to model Business Processes has been communicated in International Conference on Advanced Information Systems Engineering in 2015 [46] and extended results have been submitted to the IEEE TSC. Furthermore, we have also applied our model to different industrial projects related to the monitoring of SLA accomplishment.

In general, we follow this approach in each Contribution (Chapters §4, §5 and §6) to introduce and discuss our work.

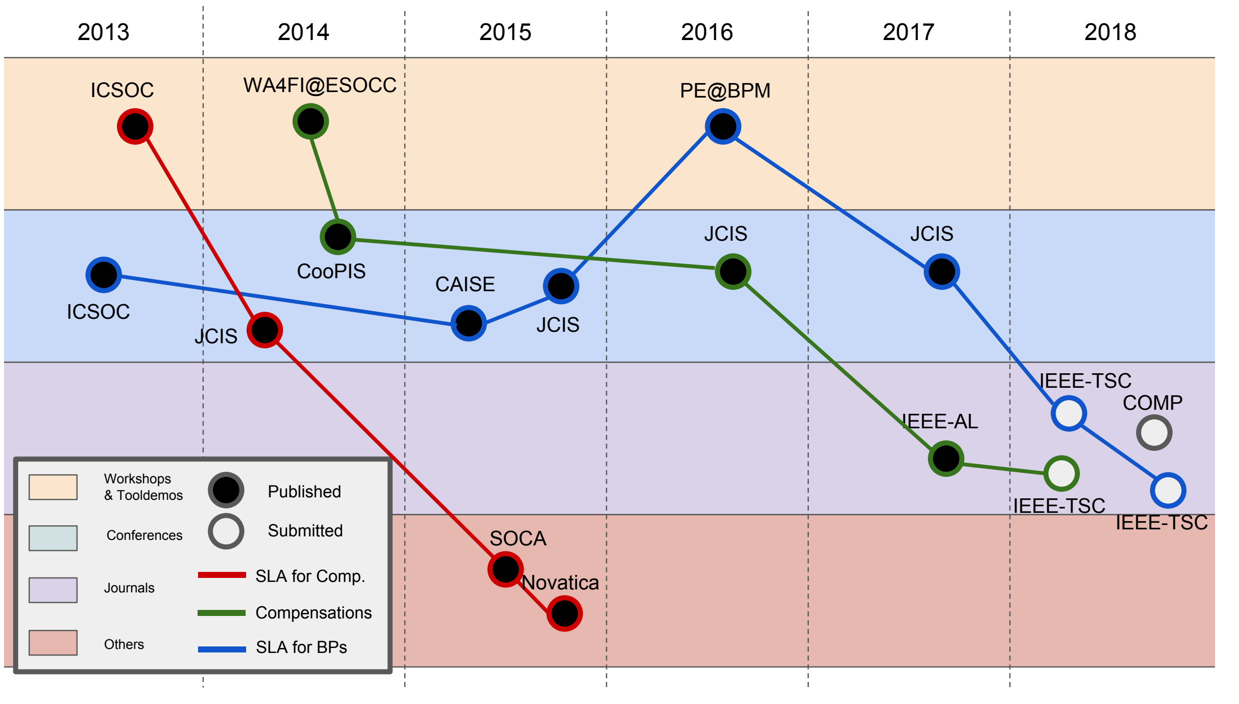


Figure 1.2: Publications Map

## 1.5 Publications

Our research work in this dissertation has followed a clear path of communication of results. The preliminary results have been communicated in Workshops and national conferences and then to relevant conferences of the area, such as CooPIS, CAiSE or ICSOC. The main contributions have been also submitted to high impact journals, although the PhD time window has not allowed to receive reviews in some cases. As this dissertation proposes the integration of previous research, the contributions are mainly the result of collaborations with researchers in different areas (from Business Processes and SLAs). Furthermore, some research stays have been performed in the context of the thesis which have started a number of collaboration whose results are also pendant of publication.

Additionally, as result of studying different adjacent areas, some contributions have been proposed out of the scope of this dissertation but during the PhD period, which are added at the end of the contribution list.

### 1.5.1 Publications Supporting this Dissertation

The Figure §1.2 depicts the map of publications developed during this dissertation.

The blue track follows the main contributions related to the management of SLAs of business processes, while the red track follows approaches related to SLAs of computational services and the green track are related to the collaborations in the analysis of compensations functions in SLAs both of computational and non computational services.

### International Journals

C. Müller, P. Fernández, O. Martín Diaz, A. M. Gutiérrez, M. Resinas, A. Ruiz Cortés. "Specifying Compensations with WS-Agreement. IEEE Latinamerica Transactions". Volume 15. Issue 7 (2017): 1335 - 1341.

C. Müller, A. M. Gutiérrez, P. Fernández, O. Martín Díaz, M. Resinas, A. Ruiz Cortés. "Automated Validation of Compensable SLAs". Submitted to IEEE Transactions on Service Computing in January 2018.

A. M. Gutiérrez, A. Del Río Ortega, A. Durán Toro, M. Resinas, A. Ruiz Cortés: "Outsourcing of Business Processes driven by SLAs". Submitted to IEEE Transactions on Service Computing in April 2018.

A. M. Gutiérrez, M. Resinas, P. Fernández, A. Ruiz Cortés: "Management of Frame Agreements". Submitted to IEEE Transactions on Service Computing in April 2018.

### International Conferences

A. M. Gutiérrez, C. Marquezan, M. Resinas, A. Metzger, K. Pohl, A. Ruiz Cortés, "Extending WS-Agreement to Support Automated Conformity Check on Transport & Logistics Service Agreements", Submitted and Accepted in Proc. of the 11th International Conference on Service Oriented Computing (ICSOC), Berlin, Dic. 2013.

C. Müller, A. M. Gutiérrez, M. Resinas, P. Fernández, A. Ruiz Cortés, "iAgree Studio: A Platform to Edit and Validate WS-Agreement documents", Proc. of the 11th International Conference on Service Oriented Computing (ICSOC), Berlin, Dic. 2013.

C. Müller, A. M. Gutiérrez, O. Martín Díaz, M. Resinas, P. Fernández, A. Ruiz Cortés. "Towards a Formal Specification of SLAs with Compensations". OTM Conferences 2014 (CooPIS): 295-312.

A. Del Río Ortega, A. M. Gutiérrez, A. Durán Toro, M. Resinas, A. Ruiz Cortés. "Modelling Service Level Agreements for Business Process Outsourcing Services". 27th International Conference on Advanced Information Systems Engineering. CAiSE 2015: 485-500.

A. M. Gutiérrez: "Advanced Analysis of Service Level Agreements". Doctoral Consortium in Service Oriented Computing and Applications. SOCA 2015: 255-258.

### International Workshops

C. Müller, A. M. Gutiérrez, M. Resinas, P. Fernández, A. Ruiz Cortés: "Towards Compensable SLAs". ESOCC Workshops 2014: 31-38

**National Conferences**

A. M. Gutiérrez, M. Resinas, A. Ruiz Cortés. "Towards the user-centric analysis of the availability" in IaaS. 10 Jornadas de Ciencia e Ingenieria de Servicios (JCIS). 2014 (received the Best Paper Award)

A. M. Gutiérrez, M. Resinas, A. Del Río Ortega, A. Ruiz Cortés. "On the Calculation of Process Performance Indicators". 11 Jornadas de Ciencia e Ingenieria de Servicios (JCIS). 2015

C. Müller, P. Fernández, O. Martín Diaz, A. M. Gutiérrez, M. Resinas, A. Ruiz Cortés. "Supporting Compensations with WS-Agreement". 12 Jornadas de Ciencia e Ingenieria de Servicios (JCIS). 2016 [received the Best Paper Award].

### 1.5.2 Further Publications

J. A. Parejo, A. M. Gutiérrez, P. Fernández, A. Ruiz Cortés, "FAST-SE: An ESB Based Framework for SLA Trading ", Proc. of the 7th International Conference on Service Oriented Computing (ICSOC), LNCS 5900: 643-644, Sweden, Nov, 2009.

A. M. Gutiérrez, J.A. Parejo, P. Fernández, A. Ruiz Cortés. "WS-Governance Tooling: SOA Governance Policies analysis and authoring", Proceedings of IEEE International Symposium on Policies for Distributed Systems and Networks. 2011.

A. M. Gutiérrez, P. Fernández, M. Resinas, A. Ruiz Cortés. "Hacia un análisis centrado en el cliente de la disponibilidad en IaaS". Revista de la Asociación de Técnicos de la Informatica. Novatica 2015

A. M. Gutiérrez, M. Resinas, A. Ruiz Cortés. "Redefining a Process Engine as a Microservice Platform". Business Process Management Workshops 2016: 252-263

A. M. Gutiérrez, F. Massena, C. Capelli, M. Resinas, F. Santoro, A. Ruiz Cortés. "Modelling Citizen Letters for Public Services automation". 13 Jornadas de Ciencia e Ingenieria de Servicios (JCIS). 2017

Jose A. Galindo, D. Benavides, P. Trinidad, A. M. Gutiérrez, A. Ruiz Cortés: "Automated analysis of feature models: Quo vadis?". Submitted to Computing in April 2018.

### 1.5.3 Intellectual Property

P. Fernández, J. A. Parejo, A. M. Gutiérrez, A. Ruiz Cortés. Tool Patent FAST, a Framework for Automated Service Trading.

## 1.6 Outline of this dissertation

**Part I: Preface.** It comprises this introductory chapter, in which we have introduced the research context of the work developed necessary to understand the content of the dissertation, we have detailed the drawbacks of existing proposals, and we have summed up our contributions to overcome drawbacks identified. Finally, we have described the context in which the thesis has been developed and the supporting publications.

**Part II: Background Information.** It provides specific information about the elements involved in the scope of the research context of our work. Specifically, Chapter §2 presents the BPM lifecycle and aspects related to its enactment and performance evaluation in the dissertation. Afterwards, in Chapter §3 the concepts related to SLAs that are handled in this dissertation are introduced together the SLA lifecycle and an study of the existing proposals regarding to support any stage in this lifecycle.

**Part III: Our Proposal.** This is the core of the dissertation and it is organised in three chapters. First of all, in Chapter §4 we propose a model to describe SLA for Business Processes. This proposal relies on WSAgreement [6], which provides the general SLA structure, BPMN [138], which is used to model the business process related to the service, PPINOT [41], which allows the definition of metrics, and iAgree [126], which provides a language to define SLOs and penalties. In Chapter §5, we introduce an architecture to support the SLA lifecycle, using our previous proposed model. In Chapter §6 we propose a model for frame and specific agreements together with a catalogue of analysis operations to support their evaluation regarding thier hierarchy.

**Part IV: Final Remarks.** Chapter §7 concludes the dissertation with a summary of the contributions and their added value, a definition of the limitations and the future work identified.

**Part V: Appendices.** Two appendices have been attached to this dissertation to complement the content of this dissertation. In particular, Appendix §A includes the formal analysis of the validity of compensable functions related to the SLA Guarantees, and Appendix §B contains an incipient proposal to optimize the management of human resources based on the service performance regarding its SLA. Lastly, a list with the complete examples of SLAs used in dissertation is also provided as Appendix.

# Part II

# Background

# Business Process Management

*A goal without a plan is just a wish*

*Antoine de Saint-Exupery (1900 - 1944),*

## 2.1 Introduction

Business process management (BPM) focuses on improving the organization performance through the definition, development and optimization of the business processes. Processes are seen as a principal asset to manage business. The business process management approach is adopted by companies as part of the quality management to improve their performance. Furthermore, in business outsourcing, the services that the organization provides to customers are seen as the outcomes of the activities performed in processes. The analysis of the involved activities helps to identify the most valuable steps, increase quality or save time [27]. Business process management usually involves the use of supporting technology, such as BPMS or other kind of enacting software. BPM approach is increasingly adopted by industry and it also discussed in academia articles focusing mainly on two viewpoints: resources and/or technology. The goal of this chapter is to provide an overview of the major concepts of BPM, focusing on those aspects that are most relevant and useful for the purpose of this dissertation.

## 2.2 Business Processes

The concept of business process is traditionally defined around the concepts of tasks, related human resources and outputs, arising from job scheduling problems in the early 20th Century. Weske [191] describes: "The formal definition and technical modeling has evolved from the process orientation trend of the 1990s, where a new way of organizing companies on the basis of Business Processss (BPs) was proposed" . Hammer et al. define a business process as a collection of activities that take one or more kinds of input and create an output that is of value to the customer [85]. However, this definition does not consider any relationship or constraint between this collection of activities, but Davenport does it in [37], where he defines a business process as "a set of logically related tasks performed to achieve a defined business outcome for a particular customer or market", emphasizing how processes can be supported by information technology.

Based on these definitions, Weske defines a business process as "a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations" [191]. A process is a set of activities related to each other targeted at the same goal. This may be the simplest and most straight definition of a process. Consciously or not, we participate in plenty of processes daily. For instance, cooking a recipe is a process in which a set of ingredients have to be cooked and mixed in a specific order to obtain the desired dish. When the process is performed in the scope of an organisation, it is called business process (BP). Thus, a BP is a collection of activities that are executed in a logical order along time to achieve a defined goal within an organisational and technical environment. To do so, they take one or more kinds of input and create an output that is of value to the customer or the market [37, 85, 191].

As stated by van der Aalst et al. [181] and Decker [39], the basis of BPM is the explicit representation of BPs. The modelling of activities is performed by business domain experts who have a specialized knowledge of the business processes. Representing a BP helps to discover weaknesses related to the order in which activities are performed, the information that is handled, and any other issue involved in BP execution. Hence, an easy-to-understand-and-use, editable, and executable mechanism to represent BPs is convenient. BP models are defined for that purpose, that is, a business process model is the representation of the activities, documents, people and all the elements involved in a BP, as well as the execution constraints between them [191]. It serves as a starting point to be analysed and improved before, during and after execution. The use of technology to manage BPs focuses on the automation of tasks together with supporting the human interaction. Current techniques enable the derivation of process models automatically from enterprise logs with process mining tools.

As simple example of BP we propose the simplified process of developing an outsourced work order by a software development company depicted in Figure §2.1 with Business Process Modelling Notation (BPMN). The goal is to develop and deploy a work order (or deliverable) with some requested features. In this example we can see the set of activities that allows realizing such a goal. The core elements in a BP are the activities and their execution order. However, there are other elements also involved in BPs, which must be also considered when designing and modelling the processes. These elements are called business process perspectives or business process dimensions. There are typically 5 different perspectives in BPs [39, 57, 191]. In this example we can identify examples for the five perspectives of BPs. (1) The functional dimension describes the activities to be performed in a BP: Plan, Develop, Deploy or Billing are part of the functional dimension. (2) The behavioural dimension specifies

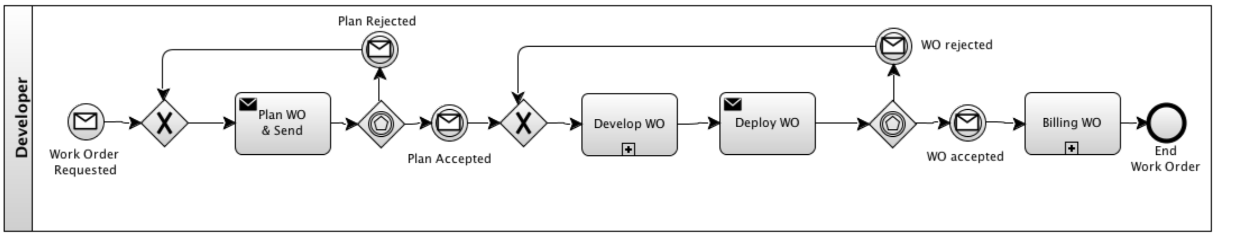


Figure 2.1: BPMN model for Work Orders Management

the control flow dependencies between these activities, e.g. the Work Order has to be planned and the plan accepted before the development starts. (3) The organisational dimension focuses on the people, roles or organisational units involved, e.g. the team responsible for development or the project manager responsible for planning. (4) The informational dimension defines the information that must be produced or consumed by activities, i.e. the data flow, e.g. the Work Order plan or the billing info. (5) Finally, the technical dimension makes reference to the different tools or machines that may be required in order to perform certain activities, e.g. the activity of deploying the Work Order to the customer infrastructure is not feasible nowadays if no computer and internet connection are available.

### 2.2.1 Business Process Modelling

Apart from BPMN, depicted in previous figure, there are other notations that allow the definition of BP models. The basis of business process management is the explicit representation of business processes, since it helps to discover weaknesses in the current organisation of activities and serve as starting point to be analised and improved. Furthermore, process documentation serves educational purposes so new employees entering the organization can quickly take up how things are done, or during organizational change programs, it can be shown how activities should be carried out in the new way [39].

This explicit representation of business processes is accomplished during the design and analysis phase of the BPM lifecycle and it is also called business process modelling. It has a long tradition and consequently several research directions can be identified. The most prominent one is related to graphical modelling notations. Between them we can highlight Eventdriven Process Chain (EPC), Unified Modeling Language (UML) Activity Diagrams, Business Process Execution Language (BPEL) and Business Process Model and Notation (BPMN). All of them have in common that they support the specification of a process control flow, defining activities, decision points with alternative paths of execution, exception handling, event handling and additional rules and constraints [176].

EPCs [163] are part of a holistic modelling approach called the ARIS (Architecture of Integrated Information Systems) framework, that defines a number of views which are similar to the dimensions previously defined for BPs, namely: the functional view (enterprise goals and subgoals and their relationships), the organisational view (enterprise organisational structure and its instanciation), the data view, the output view (outcome of BPs, i.e. products and services), and the control view, that integrates all the previous views and use EPCs to describe BPs [191].

UML [159, 160] is a language proposed by the OMG for the object oriented visual modelling. It is especially focused on the development of software systems. It provides several types of diagrams that can be used in conjunction with activity diagrams (class diagrams, component diagrams, sequence diagrams and use case diagrams between others). Activity diagrams allow to capture the process's control flow requirements by depicting what activities are performed, in what order and under what conditions. These activities are triggered by events as well as generate new events, which can be described using state chart diagrams.

Apart from these modelling notations, there exist other more formally oriented modelling languages, that allow the verification of BP formal properties such as correctness or completeness. One example is simple finite state automata [91, 107], with which processes are described as devices that maintain the state of something at a certain time and can alter this state in reaction to input as well as cause an action or output as a result of a changing state. Petri nets [3, 131] offer another graphical technique, and are a special form of graphs constituting of places, transitions, directed arcs and tokens. Places are connected to transitions via directed arcs and vice versa. Places contain tokens, which may represent signals, events, conditions, and so on. Transitions are fired through the presence of tokens in their in-place(s). As a result, the distribution of tokens is changed [176]. Petri nets present some limitations when modelling complex BPs, in these cases Workflow nets are used [2]. Workflow nets are an extension of petri nets with concepts and notations that ease the representation of BPs, like the possibility for tokens to carry information about the process instance they belong to. One of the main drawbacks of all these approaches is that they require expert knowledge to be used.

### 2.2.2 Business Process Management Lifecycle

As stated before, the basis of BPM is the explicit representation of business processes. Nevertheless, BPM also comprises other activities as described in the BPM lifecycle presented in this section. In the literature there is no consensus about the number and the name of the phases in the BPM lifecycle. They vary depending on the granularity for identifying the phases and the way of grouping the functionality in the different phases [132, 176, 181, 191, 192]. In this dissertation we will present the BPM lifecycle described by Weske in [191]. He proposes the four-phase BPM lifecycle depicted in Figure §2.2.

This lifecycle starts with the design and analysis phase. If no process exists, the goal of this phase is to define a new one; but if there is already an existing process, then the goal is to create an alternative for the current process. The new process need to be identified based, in the first case, on surveys on the organisational and technical environment, and in the second case, on the identified improvement possibilities. In either case, the informal business process description is translated to a particular business process modelling notation (usually a graphical one). Once a BP is defined, it needs to be validated to check whether all valid process instances are reflected by its corresponding business process model. Furthermore, simulation techniques can help during the validation by allowing possible undesired execution sequences to be detected and also verifying that the process actually exposes the desired behaviour. Finally, verification techniques allow to check correctness properties.

Once the business process model is designed and analysed, it needs to be implemented. This is carried out during the configuration phase in a number of different ways. If a set of policies and procedures that the employees of the enterprise need to comply with are used to implement it, no system is required. Otherwise, if a dedicated software system is needed, it must be selected and configured in order to take into account the interactions of the employees with the system and the integration with existing software systems. This integration with existing systems may involve some implementation work, for instance to attach legacy system to the current BP management. Finally this configuration must be tested, where traditional testing techniques from the software engineering area can be applied, and deployed in its target environment.

Next phase is the enactment, that encompasses the execution of the business process. On the one hand, a correct orchestration is necessary for the business activities to be performed according to the business process's execution constraints. On the other hand, process monitoring is an important mechanism for providing information about the statuses of running business process instances (BAM techniques [53] are used for this purpose). During this phase, valuable execution data is gathered. Typically, execution logs are used to orderly storage information about processes such as the start or the end of activities.

Finally, the evaluation phase uses information collected to evaluate and improve business process models and their implementations. Techniques from the fields of business process intelligence [77], and hence, process mining [182, 184], data warehousing and classical data mining [185] are applied in this phase.

Note that there not exists a strict temporal ordering in which these phases need to be executed; incremental and evolutionary approaches involving concurrent activities in multiple phases are, thus, common.

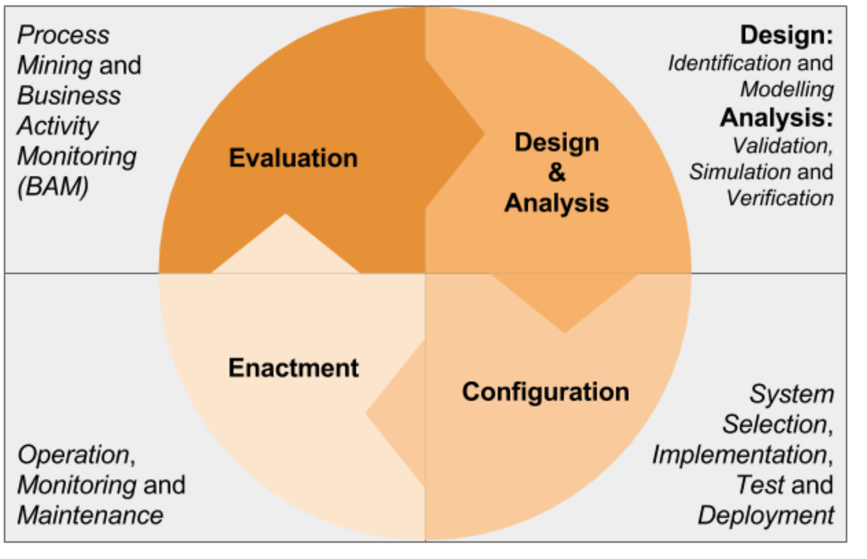

Process Mining and Business Activity Monitoring (BAM)
Design: Identification and Modelling
Analysis: Validation, Simulation and Verification
Evaluation
Design & Analysis
Enactment
Configuration
Operation, Monitoring and Maintenance
System Selection, Implementation, Test and Deployment

Figure 2.2: Business process management lifecycle as described by Weske in [191]

## 2.3 Process Performance Indicators

In order to evaluate the efficiency and effectiveness of BPs, quantifiable metrics are defined that measure the data generated by process flow. These metrics are commonly named Process Performance Indicators (PPIs). They are aimed at process controlling and continuous optimisation [28]. Often, the terms PPIs and KPIs (Key Performance Indicators) are used interchangeably, although, there is no consensus in the literature regarding the relationship between PPIs and KPIs. Some authors do not establish any difference between them [121, 148], while others consider PPIs as a particular case of KPIs, i.e. process-related KPIs [48, 193]. Finally, there are others who give different definitions to each one, placing them at different levels, KPIs nearest to the tactical and strategical level, while PPIs nearest to the operational level [28]. For this dissertation we consider PPIs as a particular case of KPIs defined for measuring the performance of BPs.

For the given process in previous section, a set of six PPIs was defined. They were defined in natural language and collected in a table (an excerpt is shown in Table §2.3). Like other KPIs, PPI definitions are recommended to satisfy the SMART criteria [51, 118, 165]. SMART is a mnemonic that broadly conforms to the following words: Specific, it has to be clear what the PPI exactly describes; Measurable, it has to be possible to measure a current value and to compare it to the target one; Achievable, it has to be able to achieve the target set in the PPI; Relevant, it must be aligned with a part of the organisation's strategy, something that really affects its performance; and Time-bounded, a PPI only has a meaning if it is known the time period in which it is measured. The definition of SMART PPIs requires a detailed specification of several characteristics of a PPI such as how it is measured or which is its target. There are a number of PPI languages, but for the purpose of this research we have chosen PPINOT

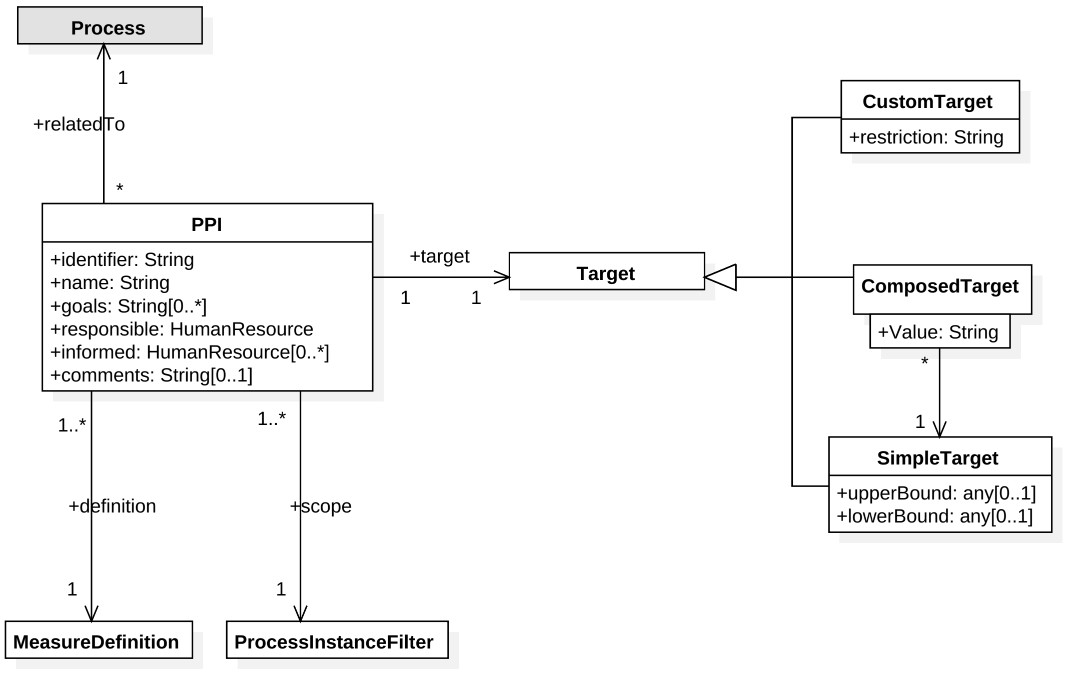


Figure 2.3: PPINOT Class Diagram

[41] because of its expressiveness and its traceability with BPMN models. Furthermore, PPINOT has been used at the core of a software tool called the PPINOT Tool Suite [42], which includes the definition of PPIs using either a graphical or a template-based textual notation [44], their automated analysis at design-time, and their automated computation based on the instrumentation of open source BPMSs. Specifically, metrics are defined using PPINOT measure definitions. As described in [41], they can be classified into three main categories depending on the number of process instances involved and the nature of the measure: base measures, aggregated measures, and derived measures.

The full model for PPI is depicted in Figure §2.3, where we can see that a PPI is formed by a Measure Definition and a Target. The class diagram for measures is depicted in Figure §2.4, where the four base types of measures are displayed: (i) Time Measure, (ii) Count Measure, (iii) Condition Measure and (iv) Data Measure, and how these types are related to different elements in processes such as Time Instant or a State Condition. Aggregated measures handle the set of base measures for a set of instances with an aggregation function such as Average, Sum, Maximum or Minimum. And, lastly, we have the derived measures to apply simple math or logic operations over a measure.

In the Figure §2.5, we display the class model for the scope of the measure. A measure can be limited by time events, by period criteria such as monthly, daily, etc. periods.

These types can be classified according to two different dimensions (cf. Figure §2.6):

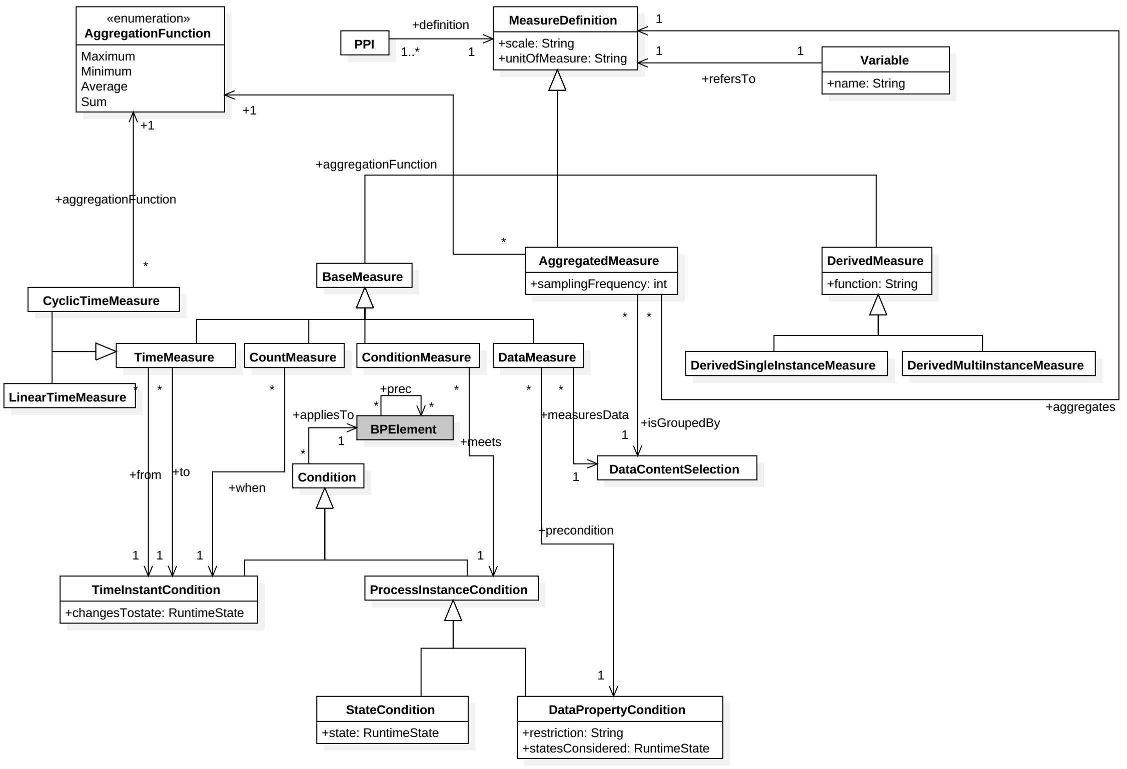


Figure 2.4: PPINOT Measure diagram

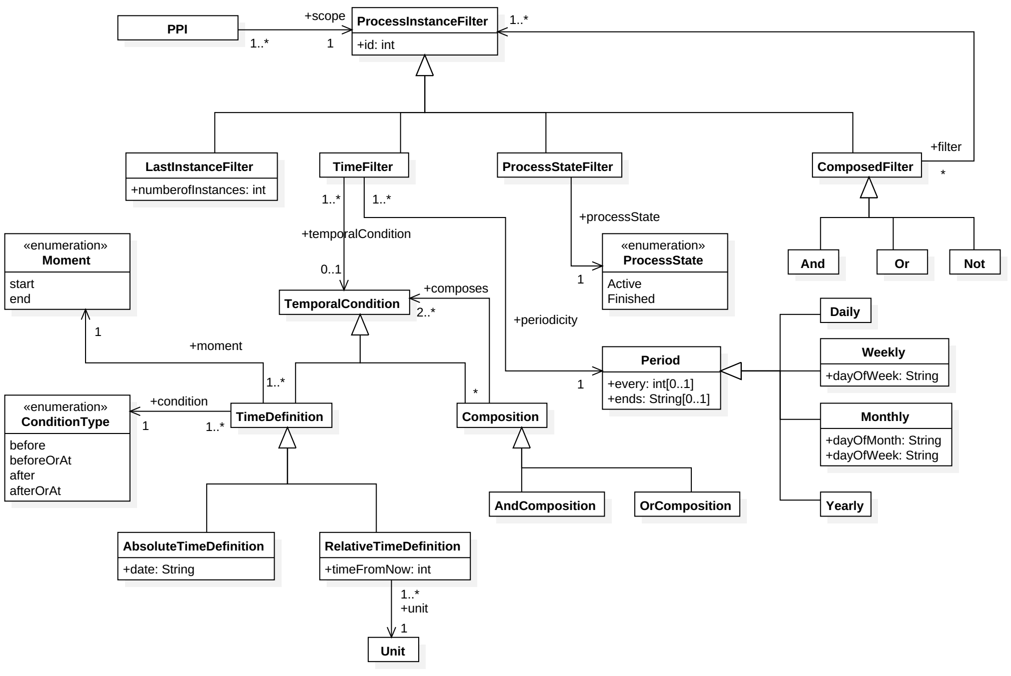


Figure 2.5: PPINOT Scopes

| | | |
|---|---|---|
| **PPI1** | Open WOs per day $\leq 4$ | Weekly |
| **PPI2** | Average time of WO Planning $\leq 1$ Working Day | Weekly |
| **PPI3** | Average time of WO Resolution $\leq 4$ Working days | Weekly |
| **PPI4** | Average time of WO Billing $\leq 2$ Working days | Weekly |
| **PPI5** | Number of WO deploy rejections per WO $\leq 2$ | Weekly |
| **PPI6** | Number of simultaeneous WO $\leq 10$ | Weekly |

Table 2.1: Committed PPIs for the Work Order process

the number of process instances necessary to compute the measure value, and the nature of the measure. As for the former, the measure definitions are single-instance measures if a single process instance is used to take the measure, and multi-instance measures if the measure value is calculated using a set of process instances. As for the latter, measures can be classified as: time measure to reflect the duration between two time points in the process, for instance, the duration of Plan WO, being the instants the start and the end of the activity; count measure to count the number of times certain condition is satisfied, such as the number of times that a WO is rejected; condition measure to check if certain condition is (for running instances) or has been (for finished instances) met, e.g. if Plan is accepted; data measure to take the value of a property of certain data object, for instance, the cost of WO billing; and derived measure, when it is calculated by performing a mathematical function over any number of measures previously defined, e.g. the percentage of the duration of activity Plan WO out of the duration of the whole process. Consequently, according to these dimensions, base measures group all the single-instance measures that can be calculated without using any other measure. They include time, count, condition and data measures. Aggregated measures are multi-instance measures calculated by applying an aggregation function on different instances of a single instance measure. Finally, derived measures, as stated above, are those whose value is calculated by performing a mathematical function over other measures. Derived measures can be both, single-instance and multi-instance measures, depending on the measures used to calculate its value.

Note that aggregated measures and derived measures are adjoining in Figure §2.6 because a derived multi-instance measure of an aggregated measure and an aggregated measure that aggregates a derived single-instance measure are two different types of measures. For instance, let $\Delta_t POW$ be the duration of activity Plan Work Order, and $\Delta_t P$ be the duration of the whole process, a derived multi-instance measure of an aggregated measure could be defined as $\frac{max(\Delta_t POW)}{max(\Delta_t P)}$, whereas an aggregated measure of a derived single-instance measure could be something like $max(\frac{\Delta_t POW}{\Delta_t P})$.

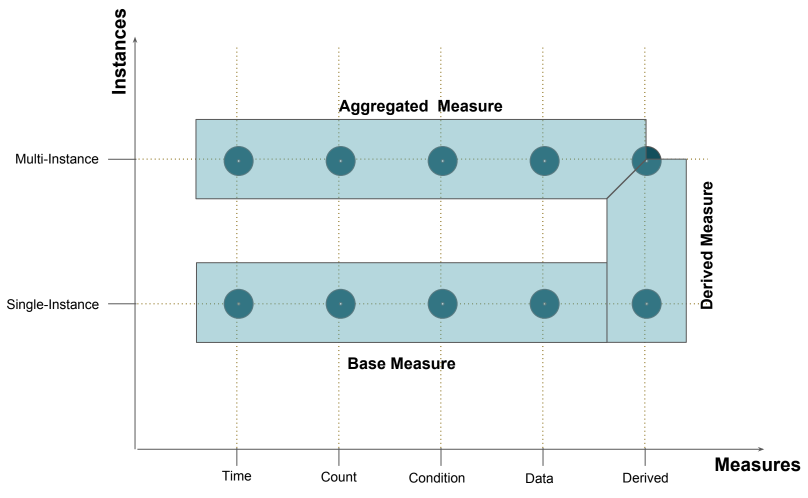


Figure 2.6: PPINOT Dimensions according to [41]

## 2.4 Process-Aware Information Systems

The phases of the BP lifecycle are increasingly supported by software aware of the executing process. The enactment of different BP has been traditionally supported by software tools, such as bug tracking tools in software development or purchase tracking in Enterprise Resource Planning tools (ERP). These systems also facilitate the definition or monitoring of BPs. Dumas et al. introduced the concept of Process-Aware Information System, henceforth referred to as PAIS [54]. This system is defined as "a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models" [189].

Weber et al. [189] used a PAIS architecture in order to discuss the different perspectives of PAIS. This architecture can be viewed as a 4-tier system:

- Persistence Layer enables the necessary support for a database management system to maintain the data persistence.
- Application Layer is responsible for storing the application codes and implementations of the various functionalities of the activities. These implementations can be owned by different organizations.
- Process Layer runs the process logic. In particular, it contains the schema and complete specification of the process model which is used for the process execution.
- Presentation Layer provides different build- and run-time tools for customers, e.g., a process template editor and an application program interface that enables the different components to be monitored.

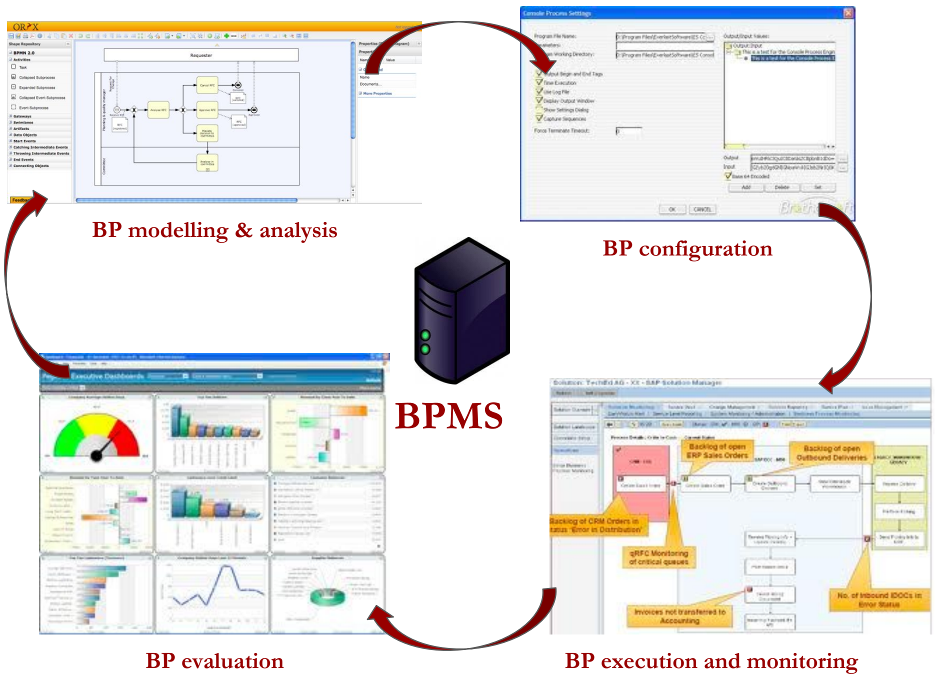


Figure 2.7: BPMS support

When a PAIS handles explicit BP models created with a generic BP language are named Business Process Management Systems. Open Source BPMSs are YAWL [64], jBPM [1], Activiti jBPM [2] or Camunda [3]. Others, such as Architecture of Integrated Information Systems (ARIS) [4], IBM Business Process Manager [5] or AuraPortal [6], are commercial. They are composed of a number of subsystems that address functions belonging to the different phases in the BP lifecycle. An overview of the typical architecture of a BPMS is depicted in Figure §2.7. In the following we provide definitions for every subsystem (inspired by the descriptions given by Weske [191]), specifying the BP lifecycle phase to which each it is related.

- Modelling and Analysis. A BP modelling tool is used for creating BP models, in which all the BP perspectives (cf. Section §2.2) involved in a process should be properly represented. The BP models can be then used by process analysts for analysis purposes. Thus, after modelling, the role of the BP analysts comes on stage. They deal with the more tactical aspects of BPM (discovering, validating, documenting and communicating BP-related knowledge) and typically do pro-

[1]http://www.jbpm.org/

[2]https://www.activiti.org/

[3]https://www.camunda.org/

[4]http://www2.softwareag.com/corporate/products/aris_alfabet/bpa/aris_architect/default.aspx

[5]https://www.ibm.com/us-en/marketplace/business-process-manager

[6]https://www.auraportal.com

cess and data analysis, make changes to processes, and make sure that any ramifications downstream and upstream from the process have been checked over. BP models that are created with the BP modelling tool are stored in a BP model repository.

- Configuration. When processes and system are prepared, the BP environment triggers the instantiation and enactment of BP instances based on the BP models. The BP architects may also participate in the preparation of the system during the Configuration phase of the BP lifecycle.

- Enactment. The Process Engine is the core component of a BPMS. It is responsible for instantiating and controlling the execution of BPs. It orchestrates the execution of activity instances, calling entities that act as providers of the required functionality. Typically, these providers are human resources of the organisation (i.e. people), or service providers (in service-oriented architectures). In the case of having human participants, an activity typically generates one or more work items (a.k.a. tasks) which together constitute the work to be undertaken by a user (or a group of users). The list of work items associated with a given participant (or with a group of people) constitutes his/her (their) worklist. The work item(s) are normally presented to the user via a work-list, which maintains details of the work items allocated to a user, and a worklist handler, which interacts with the work is on the behalf of the user [7]. The event logs (a.k.a. history logs, or execution logs) are used to store the information generated during BP execution, which is necessary to perform runtime and post-mortem analysis.

- Evaluation. Dashboards can be generated as a result of the analysis (specially design-time and post-mortem analysis) with the aim of studying the actual behaviour of a BP in order to identify and correct potential problems (at design time), or define improvement mechanisms to be applied in subsequent executions. The BP analysts are usually also in charge of performing such evaluations. It may have four main goals: (i) to check whether the process instances worked as expected, and detect unexpected behaviour of the process under certain circumstances. This check is related to the degree of compliance of the process with the rules that must be fulfilled in the organisation; (ii) to find out bottlenecks or conflicting activities; (iii) to perform any kind of statistical analysis over one or more BP perspectives; and (iv) to identify changes that lead to an improvement of the process. Techniques from the fields of BP intelligence [77], and thus, process mining [177, 182, 184], data warehousing and classical data mining are usually applied in this phase.

[7] http://www.wfmc.org/

## 2.5 Summary

In this chapter, we have introduced the main concepts related of BPM. In particular we have provided definitions for BP and BP Management Systems. We have also described a notation for BP performance indicators, which is relevant for the context of this dissertation. Finally, a brief description of other tools related to process automation have been provided.

# SLA Management

*Let each man exercise the art he knows.*

*Aristophanes ( 427 BC - 386 BC ),*

## 3.1 Introduction

In scenarios where a party consumes services from a service provider under a previous agreement, the service consumer usually requires some expected performance on the services they intend to consume. These expectations are defined over quality or non-functional properties that can be measured so both parties can check if the service fulfills them. These measures and fulfillment conditions are defined as service level objectives (SLOs). The service description and SLOs, are included in a Service Level Agreement (SLAs) agreed between consumer and provider. The evaluation of SLA fulfillment is a key part to enforce service quality. A number of research efforts have focused on proposing models for SLA management in computational and non-computational domains.

In the industry, most of the SLAs are described by means of natural language and are intended to be used only by humans for legal procedures. With the rise of service oriented architectures, a number of research proposals have defined different formal languages with the aim of analyzing some properties of the SLAs [18, 31, 158] or enabling the interoperability between consumers and providers easier [6, 111, 135]. Among the approaches of the latter group we find the WS-Agreement [6] recommendation that provides an SLA specification based on a XML language to describe domain-independent agreements, offers and templates. The recommendation also provides a lifecycle for the agreements, including the creation, validity stages and the different reason to finish an agreement (agreed period, violation or cancellation). It lacks of a language to provide semantics to the service description and SLOs but is extensible to achieve a fullyfledged language to describe the service functionality and the SLOs. One of the proposals to fill this gap is iAgree, a human-readable language based on WS-Agreement. In this chapter we explain in detail the WS-Agreement structure together with iAgree and a full tool suite to manage SLAs, Governify. In addition, we also short review the different scope and elements supported by other SLAs specifications.

## 3.2 WS-Agreement and iAgree

WS-Agreement is a specification that describes computational service agreements between different parties. It defines both a protocol and an agreement document metamodel in the form of XML schema [6]. The general structure of WS-Agreement documents is depicted in Figure §3.1 using a class diagram to clarify the XML schema proposed in the recommendation [6] for the documents. Some of the elements, such as *ServiceDescriptionTerm*, *ServiceLevelObjective* or *Constraint*, can be extended with a new sublanguage to describe them. Thus, a WS-Agreement document is composed of an agreement *Context*, and agreement *Terms*. Template documents can be described to include agreement creation constraints to restrict the space of offers that can be created from them. For example, a sea carrier defines a template document for their shipments from one port to another where they describe the generic terms of the shipments, such as origin and destination port, standard transport conditions, such as container size or weight, and estimated transport duration. Any customer who contracts this carrier can use this template to make an agreement offer with specific transport dates and including penalization terms in case of not accomplishing the estimated transport duration. According to WS-Agreement the terms included in the template are not mandatory for the agreements based on it.

iAgree syntax, defined by Müller [126], is designed for making WS-Agreement documents more human-readable and compact than with the original XML syntax. All examples included in this paper are defined using iAgree. Although iAgree structure lightly differs from WS-Agreement, iAgree sections can be easily mapped to WS-Agreement structure. The class diagram for iAgree is depicted in Figure §3.2. Through next sections we describe the agreement structure referring to the example in the code in Figures §3.3, §3.4 and §3.5. This example is based on the SLA defined by Amazon Web Services for its computation service, Amazon EC2[1]. In order to know the mapping with WS-Agreement, we will refer to its sections in the corresponding iAgree description. The complete example is listed in the Appendices, Section §C.1. iAgree is evolving as it is applied in different projects. Refer to the website for the current specification: `http://iagree.specs.governify.io`.

### 3.2.1 Context

The agreement context contains information about the roles in service provision. Apart from who is the service provider and consumer, the initiator of the agreement proposal is also included, i.e. who creates the agreement offer and proposes to the

[1]`http://aws.amazon.com/ec2/sla/`

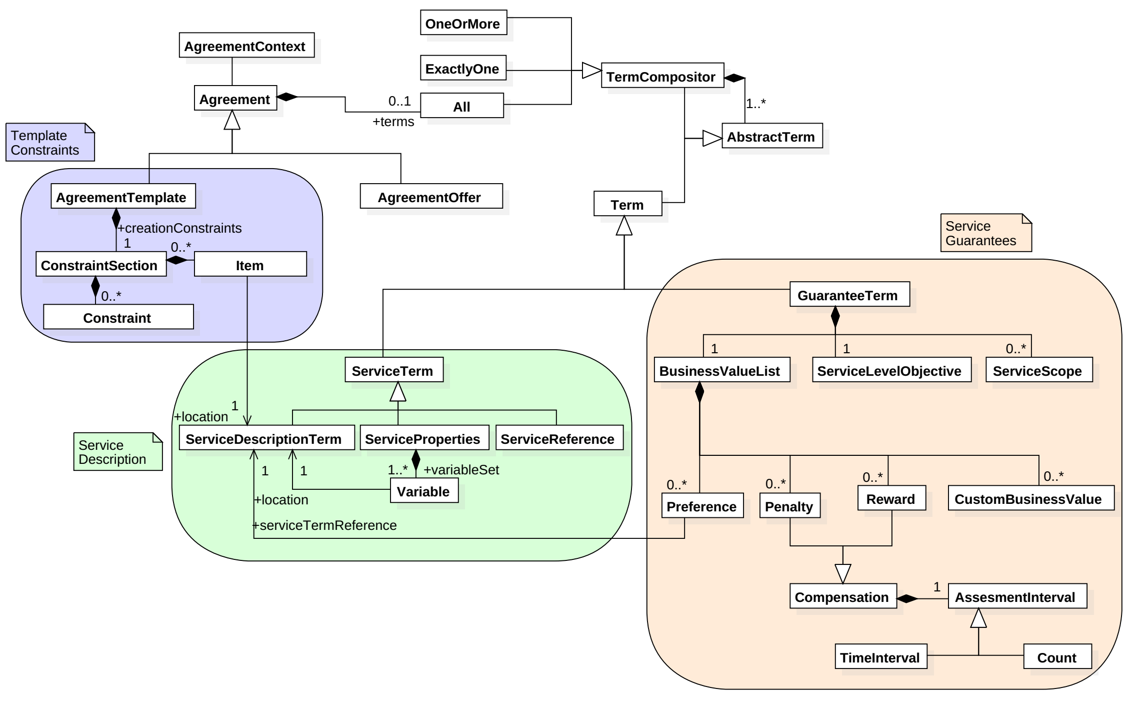


Figure 3.1: Class Diagram for WS-Agreement

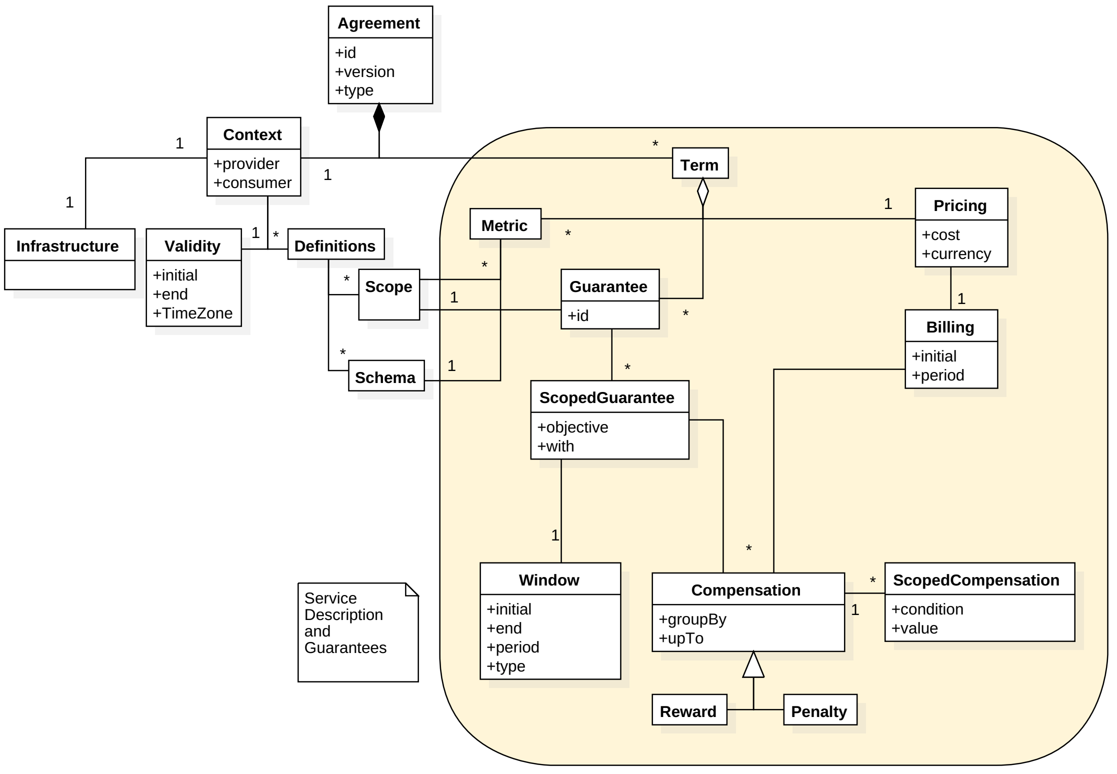


Figure 3.2: Class Diagram for iAgree

```
id: Amazon_EC2
version: '1.0'
type: agreement
context:
  provider: Amazon
  consumer: Consumer
  validity:
    timeZone: Europe/Madrid
    initial: '2018-07-13T00:00:00.000Z'
  definitions:
    schemas:
      ServiceCredit:
        description: Percentage to decrease in the next bill
        type: integer
        unit: '%'
    scopes: { region }
```

Figure 3.3: Context section using iAgree syntax

other party. The service provider can be either the agreement initiator either the agreement responder (and reverse for the service consumer). The context also includes the service description and data types definitions for non monitorable properties. Non monitorable properties are those which do not depend on the service execution or are significant for the evaluation of the service performance. The monitorable properties are defined later in the document and will be described in next subsection. For instance, in our example in Figure §3.3, Amazon, who is the provider of the computing service, plays the role of responder because it receives agreement offers from other parties. And the Amazon penalization is defined with a schema for Service Credits as a percentage value. Additionally, the scopes where the guarantees shall be parameterized are also described in the Context section. In the case of Amazon, the service billing and compensations depends on the computation region, so the region is one scope value for Amazon EC2.

In addition, it optionally provides information about the agreement parties endpoints, the agreement's lifetime or references to the template from which an agreement offer is created, if this is the case. The infrastructure object relates the SLA to the specific information systems supporting the service execution in order to provide the required execution information (e.g.: performance values). In the example in §3.3, the definition of validity periods (VPs) is included. The WS-Agreement context section is similar to iAgree but it does not include data types.

### 3.2.2 Terms

The terms section of an iAgree document describes both the characteristics of the services to be provided and the guarantees on such services by involved parties. in iAgree, the service terms are divided into pricing, metrics and guarantees.

Pricing section defines the properties to properly process the service billing such as currency, accounting period or cost. The cost depends on the service and can be measured by time units (e.g. 3€/hour) and/or other metrics (e.g. number of executed instances). Commonly, compensations are described with a percentage discount or fee over the billing, as consequence of under- or over-fulfillment of SLOs. Therefore, the percentage property is named in the pricing but the compensation definition is related to the SLO in the corresponding Guarantee.

Metrics use iAgree schemas to describe the data types of the service properties that are monitorable or computed (e.g. availability of a computation unit). A metric definition can include an endpoint to get the execution values. The values can be monitored in a specific time window (monthly, hourly, ...) and this window can be static (e.g.: natural weeks from Monday to Sunday) or dynamic (the previous 7 days from a given moment). The monitoring mechanism is not defined in the model, just the service endpoint that provides the metric. These metrics can be used to define guarantees, which are detailed in next subsection. In our example, we include the metric *Monthly Uptime Percentage (MUP)*, which measures the availability of the computing units in Amazon EC2 for a given user. It is measured by natural months according to the computing regions defined by Amazon Web Services [2].

```
terms:
  pricing:
    billing:
      period: monthly
      initial: '2017-11-12T10:35:36.000Z'
      penalties:
        - over:
            ServiceCredit:
              $ref: '#/context/definitions/schemas/...'
  guarantees:
    ...
```

Figure 3.4: Terms section using iAgree syntax

In the example in Figure §3.4, we describe the pricing term and the metrics for Amazon service. In this case, the billing is monthly required and the compensation is defined with a variable *ServiceCredit* over it. The only metric in this service is the MUP.

## Guarantee Terms

The guarantee terms (GTs) describe the SLOs that an obligated party, usually the service provider, must fulfill as part of the agreement. The SLO of a guarantee term is defined with an assertion over a monitorable variables which were described in Metrics section and over external parameters such as date, time, etc.

Each guarantee term can be parameterized with additional properties to specify the scope where it is applied such guarantee. In the Amazon computing service, they define an SLA with a single SLO. However, they organize their global computation service by regions and the billing and the context to apply the guarantee is limited to each single region (Amazon currently defines more than 15 computing regions). That is, if the objective of 99.95% is not accomplished in a region, the penalty is applied

[2] https://aws.amazon.com/es/ec2/sla/historical/

for the billing of that same region. We can parameterize the guarantee using the region as scope. Then, the guarantee is evaluated for each region value (e.g.: Canada, US East, ...). In practice, adding a scope variable with $N$ possible values to a guarantee, it is as defining $N$ guarantees (e.g.: $G1.1: MUP \geq 99.5\ AND\ region = Canada$, $G1.2: MUP \geq 99.5\ AND\ region = USEast$, etc). We have described in the Figure §3.5 the SLA from Amazon EC2, with the clause offered by Amazon to penalize itself by a service unavailability. In the clause, the customer is able to claim for a reward of a Service Credit when $MUP$ objective is not achieved. However, information such as the claiming procedure or the fact that Amazon requires that the customer proves this violation, is not supported by iAgree formalisation. In order to prove the violation the customer has to monitor and compute the MUP by subtracting from 100% the unavailable period. It is not the case of Amazon, but the SLOs can be defined different for each value of the scope variables (e.g. to define a different goal and compensation for each specific region).

```
id: Amazon_EC2
...
terms:
  ...
  guarantees:
    - id: Amazon_GT
      scope:
        region: *
      of:
        - objective: MUP >= 99.95
          scope:
            region: *
          with:
            MUP: {}
          window:
            type: static
            period: monthly
            initial: '2016-07-13T00:00:00.000Z'
          penalties:
            - over:
                ServiceCredit:
                  $ref: '#/context/definitions/schemas/...'
              of:
                - value: '10'
                  condition: MUP >= 99.00 && MUP < 99.95
                - value: '30'
                  condition: MUP < 99.00
```

Figure 3.5: Guarantees using iAgree syntax

Regarding the penalties, Rana *et al.* in [151] provide an approach to (1) identify classes of penalty clauses that can be associated with an SLA; (2) define how to specify penalties in an extension of WS-Agreement; and (3) specify to what extent penalty clauses can be enforced based on monitoring of an SLA. An extended analysis of compensations have been described during the current work and it is included in the Appendix §A.

As summary, a compensation function is usually defined as a function from the metric $M$ guaranteed in the SLO to $\mathbb{R}$ that associates a compensation to each of the values of *M*. The compensation function is usually proportional to the utility of the metric values (i.e.: Poor availability can be penalized, high availability can be rewarded). As in the SLA, both service parties, customer and provider, can commit to different SLOs, the responsible of the compensation can be either the service provider or the service customer. Therefore, the compensations have two roles, namely guarantor and beneficiary of a guarantee, and they are usually provider and customer but it can be the opposite. For example, in a cloud scenario, the availability guarantor is the cloud provider and the beneficiary is the customer.

WS-Agreement also proposes to add other service properties as Terms to describe the service as it is, not to negotiate about them and their performance. An example of these properties, namely Service Description Terms, would be origin and destination cities in the Transport Service example.

In order to simplify, the Agreement definition in iAgree, all the terms are mandatory. However, the terms in a WS-Agreement document can be grouped using three different terms compositors denoting that the comprised terms are either mandatory, optional, or alternative, as it is depicted in Figure §3.1.

- **All** terms compositor: every comprised term or compositor is mandatory. In other words, all of them must be fulfilled. WS-Agreement specification imposes that at the top level of the terms section, all terms must be inside a mandatory terms compositor. For the sake of simplicity we consider as implicit such a top level mandatory terms compositor.
- **OneOrMore** optional terms compositor: every comprised term or compositor is optional. In other words, a set of them, but at least one, must be fulfilled.
- **ExactlyOne** exclusive terms compositor: comprised term or compositor are exclusive. In other words, only one of them must be fulfilled.

Term compositors can be nested, therefore enabling the specification of alternative branches with potentially complex nesting within the agreement terms. Choices expressed using compositors can be exercised by the party that makes the next step in the agreement creation process, i.e., by the agreement initiator if it is creating an offer from a template, by the agreement responder if it is creating an agreement from an offer, or by the service provider if it is delivering the service according to a previously created agreement.

Note that parties can exercise the choice but it is not mandatory to do so. In other words, the term compositors can remain in an offer and even in a final agreement, exactly as they were defined in the former template.

Furthermore, in WS-Agreement, the guarantees can be categorised using also assertions, namely qualifying condition (QC), or a set of service operations, which express the conditions under which the guarantee holds. WS-Agreement also considers that

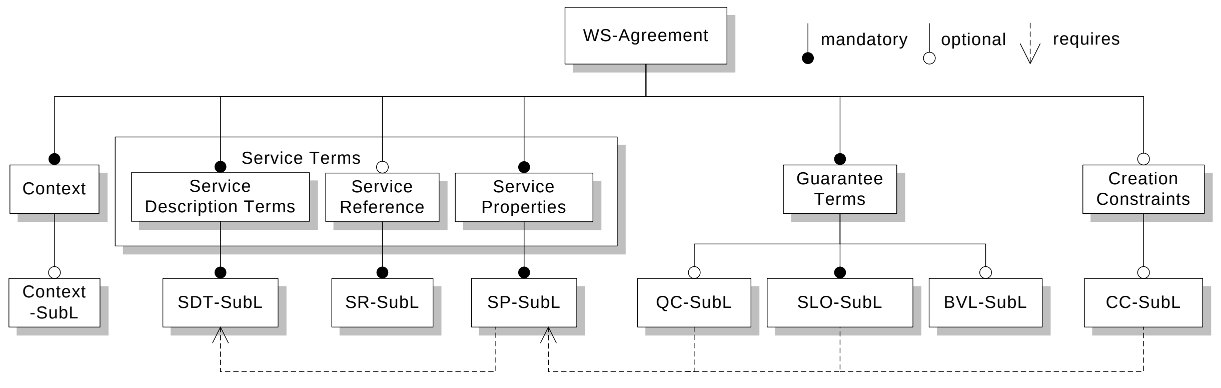


Figure 3.6: WS-Agreement and sublanguages

SLOs can be related to externally defined key performance indicators (KPIs), i.e. externally defined monitorable variables and to a list of business values (BVL) to describe the relative importance between the terms, penalties, etc.

### 3.2.3 iAgree Configurations

As WS-Agreement do not provide a specific language for service terms, different languages such as JSDL (Job Submission Description Language) and the WSLA [94] expression language, have been used for this task since WS-Agreement do not impose additional constraints on their expressiveness. Therefore, in order to create a WS-Agreement document is required to decide which are the languages that are going to be used in it. The optional and mandatory sublanguages than can be described in WS-Agreement are depicted in the Figure §3.6. On the one side, we can define a language to describe the service context, which can be domain specific. On the other side, we have different language/s required to describe the terms, which can include a language for metrics, other for service description terms, other the guarantees (including a language for qualifying conditions), another for SLOs and another for Business Value Lists, which are values added in WS-Agreement in order to the parties can express the *importance* of the SLO. This *importance* can be expressed as a priority parameter, as the confidence on the accomplishment of the SLO or as a compensation function. The specific set of languages used to describe an SLA is called a WS-Agreement Configuration (WSAC) in Müller [122]. For instance, the implementation of WS-Agreement developed by the Open Grid Forum (OGF), WSAG4J [134][3], configures a WSAC that involves using JSDL as the language for specifying service description terms and JEXL as the language for specifying service level objectives. SWAPS [139] also proposes a WSAC by using WSDL-S/OWL as language for the service functionality, and WSLA [94] expression language for SLOs.

Regarding iAgree, it provides its own WS-Agreement configuration. iAgree defines

[3]WSAG4J

service terms as attribute-value pairs. The metric language which is used to define the metric of a service property, provides the domain of the service property, i.e., it describes its data type and its allowed values. And also defines expression languages to specify SLOs and creation constraints. In this case, the expressions are described with assertions using logical, relational, and algebraic operators defined on the domain of the service descriptions, service properties and literals. In practice, in order to support different analysis operations as we will describe in Sections §3.4 and §3.5, the expression language used shall be restricted by the types of expressions that a particular solver can use as these expressions.

## 3.3 THE SLA LIFECYCLE

During the last decade SLA management has become both: (1) an increasingly important issue in the IT infrastructure [120, 170], and (2) a fundamental aspect to achieve a quality-driven service consumption. In this context, in order to ease the management of SLAs, it is necessary to identify the different phases that should be followed during the so-called SLA lifecycle to create and make use of SLAs [90].

Several proposals mention different phases of the SLA lifecycle depending on their relationship with the related service lifecycle. Thus, authors like Comuzzi [31], Radha Krishna [101] and Vonk [186] consider the early SLA phases performed before the agreement is considered firm and services executions are ruled by it. Other proposals, such as the WS-Agreement specification [6], pay more attention to such late SLA phases performed from the beginning of execution of the agreed service to the termination or expiration of the agreement. Finally, authors like Koller [98] try to consider each and every SLA management phase in its approach. In such general-purpose proposals a full SLA lifecycle is presented. Among these proposals we focus on the phases mentioned in the Fernandez-Montes PhD [60], where a model for trading architectures is presented. These phases are really shared with the proposals with limited scope introduced before. Thus, the phases we consider in this dissertation for the SLA lifecycle are the followings:

- Preparation. It is a special stage that involves the creation of a document with the service offered by the provider and the analysis of its functional and non-functional requirements by the consumer (also called preferences).
- Information. The goal of this phase, which is also called as discovery and selection, is to match service providers with potential consumers and vice versa. In other words, either consumers try to find a suitable provider for their desired services and service levels, or providers try to find service demands from potential consumers. In order to locate and select a suitable counterparty, the parties interchange public information during this phase that can be eventually aided by an intermediary party that provides an ordered list of potential counter parties [68].

- Negotiation. It is the phase in which parties try to achieve an agreement on the service level for the service provisioning. Several offers and counter-offers are usually exchanged at this complex phase in which several strategies can be followed in the pursue to achieve a final agreement signed by the parties [153].
- Deployment. In this phase both, the service provider and consumer set up a deployment plan to make it possible to follow all terms established in the agreement settled in the previous phase. The phase delimits when the service subject of the agreement can be started to be consumed or provided.
- Fulfillment. This is the last phase in the contracting process and it involves the monitoring of the whole process in order to ensure that both parties observe the obligations established in the agreement, correctly. This phase requires specific monitoring tools and approaches [113, 141, 151].
- Termination. This is the last phase of the SLA lifecycle and it delimits when the SLA expires and then, the service provisioning is stopped for the specific consumer. At this moment if the agreement includes termination clauses they must be considered.

In order to automated the operative of theses phases (detailed in subsections), the SLA is considered as a first-class citizen to analyze service performance. The most relevant aspects for these operational analysis is described in next section §3.4.

### 3.3.1 Early SLA phases in iAgree/WS-Agreement

The agreement creation protocol supported by the WS-Agreement recommendation proposes the use of agreement offers and templates to perform *Preparation*, *Information* and *Negotiation* phases of the SLA lifecycle mentioned previously.

The most common usage scenario of the agreement creation protocol starts with the *Preparation* of an agreement template. A template is a partially completed offer that specifies customizable aspects of the documents, and rules that must be followed in creating an agreement, which are called agreement creation constraints. Creation constraints (CCs) are optionally included in an agreement template in order to specify the mandatory presence of specific elements and their acceptable values for resulting offers created from such templates. To this end, the template may define specific items using XML Schema [188] to delimit the possible value assignments for the service terms or, if the creation constraint involves several elements, they can be specified using any suitable constraints language that must be described by means of one or more sublanguages. Since items can be considered as a particular case of constraints, in iAgree we just consider a sublanguage for the constraints. Therefore, an agreement offer created from such a template must observe these constraints. We detailed the SLA lifecycle in section §3.3.

Once the template is published, it can be found by any potential customer demanding the service at the *Information* phase of the lifecycle. Such a potential customer prepares an agreement offer compliant with the published template and initiates the creation process by sending it to the provider. Finally, the provider either accepts or rejects the agreement offer. If rejected, new offers may be sent to the provider in this *Negotiation* phase, but if accepted, such an offer is deployed as the agreement that regulates the service provision. As can be seen, the customer initiates the interaction and therefore it plays the role initiator in the agreement creation process, whereas the provider plays the role responder. Although the previous is the usual usage scenario, two additional considerations are included in WS-Agreement. First, the publication of a template is optional and thus, any party may send an agreement offer to other party without any template published.

However, the acceptance of an agreement offer is more likely if it is defined based on a previously published template. Second, although the service provider usually acts as responder, WS-Agreement also allows consumers to play the role of responders by publishing templates with the service they intend to consume and some desired guarantees. In these cases, providers would initiate the SLA creation process by sending agreement offers including the service they provide and their capabilities.

### 3.3.2 Late SLA phases in iAgree/WS-Agreement

Once an SLA is signed, service set ups are arranged, such as resources allocation or internal process customizing during *Deployment* phase. During service execution the parties have to provide some mechanism to check whether the service fulfills the committed service level objective (SLO) during its provisioning, that is the *Fulfillment* phase. This checking implies monitoring the service performance.

SLAs must provide the information of how SLOs and metrics must be monitored to enforce the trust by involved parties. In WSLA [111] documents by means of the measurement directives element. In turn, WS-Agreement provides a metrics for each service property that may include such a kind of measurement information, and the scope to denote which SLO must be monitored when a service operation is being providing. When an SLO is violated, the penalties and rewards included within the guarantee term apply. In iAgree, the computer attribute of each metric provides the measurement for this metric. The computer is a reference to an external service which provides the measures for a metric. It is out of the scope of iAgree how this measure is obtained. It can be queried from a database, listened from an event sequence or processed as a derivate value from other more simple metrics. The monitoring usually depends on the system which performs or supports the service execution. The former is the common case in computational service and the second in services which include manual tasks but performance evidences are captured in an information system. Therefore, defining mechanisms for measuring each metric would require consider the domain where the service belongs, the supporting systems, etc.

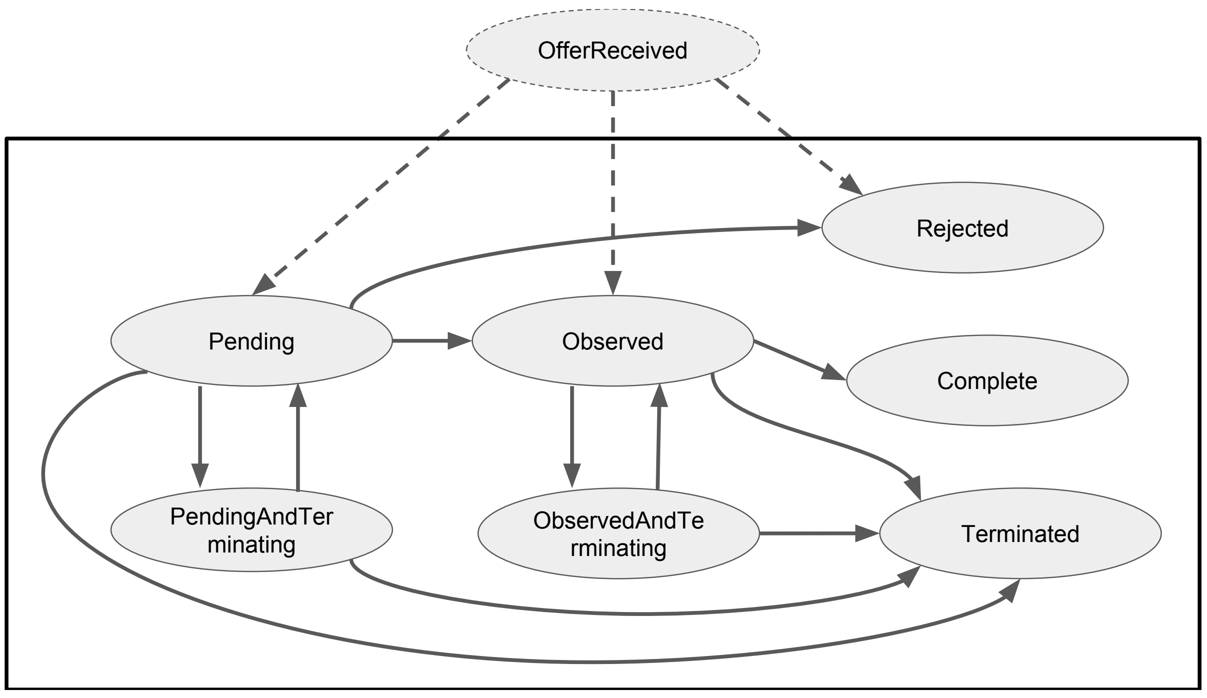


Figure 3.7: Agreement states in WS-Agreement

In order to avoid this ad-hoc dependence, WS-Agreement does not provide a technique to monitor the service in order to check the fulfillment of WS-Agreement documents, but it considers several states of the SLAs and Guarantee Terms (GTs). As depicted in Figure §3.7, an SLA in WS-Agreement can be in one of the following states: (i) Pending, (ii) PendingAndTerminating, (iii) Observed, (iv) ObservedAndTerminating, (v) Rejected, (vi) Complete and (vii) Terminated; and as depicted in Figure §3.8, the GTs can be in one of the following states: NotDetermined, Fulfilled, Violated. Such states are thoroughly described in the recommendation [6] and we do not intend to explain them in this dissertation but to remark that an SLA or a GT would change its state by the use of an external monitoring technique that provided information about the current SLA and GT fulfillment.

The monitor component included in such external monitoring technique should consider some aspects related with the monitoring information that SLAs comprise: The monitor must be aware of such a monitoring information in order to monitor specific SLOs and metrics when the scoped service operations are executed at runtime. In addition, if the guarantee terms have local periods specified they must be considered as the periods in which the terms must be monitored.

The monitor must include enough monitoring logic to obtain the term state. Such a logic depends on the complexity of monitoring and measuring the SLOs and metrics. For instance, the monitor must compute the MUP of the AmazonEC2 scenario, explained in Section §3.2, by subtracting from 100% the average of the error rates, i.e. internal server errors divided by the requests during each five minute period of

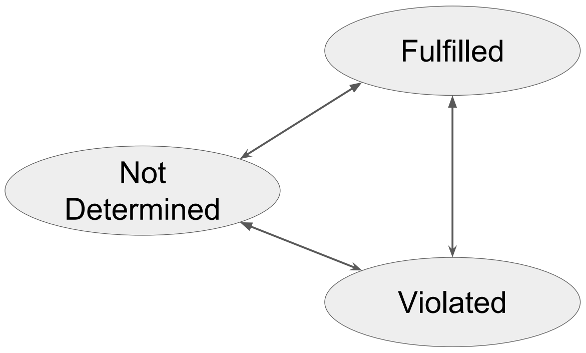


Figure 3.8: Guarantee Term states in WS-Agreement

a monthly billing cycle. The responsibility of monitoring and demonstrating an SLA unfulfillment uses to fall into the customers, or an intermediary third-party. For instance, a low MUP in the AmazonEC2 scenario must be monitored and demonstrated by the customer, as in other commercial services such as Google Cloud Storage[4], or RackSpace[5]. In the AmazonEC2 case, the customer must provide within ten business days after the end of the billing cycle in which the errors occurred, both: (1) the time instants in which it suffers the lack of service; and (2) a log including the requests and error messages provided by Amazon.

The monitoring result can be used both in proactive and reactive procedures. The proactive behaviors take actions to avoid an SLO is unfulfilled when performance levels are close to the SLO limit values or is predicted that it could happen. These actions can be related to scale computational resources, plan different human assignments or decrease other performance attributes (e.g. increase speed over quality) in order to accomplish SLO. Reactive decisions are commonly defined with compensations functions so when an SLO is unfulfilled, a penalty is automatically applied. In other cases, similar actions to preemptive ones are taken (e.g: scale resources).

When there are not explicit actions defined for unfulfilling SLOs, manual decisions has to be taken. In the worst case, the SLA is considered broken so both parties have to renegotiate a new agreement (e.g.: with less restrictive performance objectives) or finish the service interaction. Renegotiation can follow similar mechanisms to the original negotiation phase.

In the *Termination* of the relationship between provider and consumer ruled by the SLA terms, some assessments about termination arrangements such as final billing, service data ownership, etc can be performed. As it was stated, the agreement can finish because its violation, a manual decision or just because its validity period is ended. The consequences of the termination will also depend on the termination cause.

[4] https://cloud.google.com/products/cloud-storage
[5] https://www.rackspace.com/

## 3.4 SLA Analysis

In order to support the SLA phases introduced before, different analysis operations can be performed. For example, to define the SLO thresholds in the SLA, the customer can evaluate existing service execution logs to define a reasonable threshold. Or to define negotiation strategies, the provider can analyse the utility of the different terms of an SLA offer in order to accept or reject it. The PhD Thesis from Carlos Müller [122] defines an extensive catalog on possible operations over SLA documents. The origin of this dissertation is developing these operations to different scenarios where there are no existing approaches to implement them. For the sake of this work, among these operations, we highlight the following ones:

### 3.4.1 Compliance between Offer and Template

According to WS-Agreement, it is defined that "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 AgreementResponder value must match the value specified in the template; and the TemplateId must exactly match the name provided in the template document against which compliance is being checked.

Then, we interpret that there exists compliance between an offer and a template if and only if none of the offer terms contradicts the template creation constraints which is included and proposed in the PhD Thesis from Carlos Müller [122]. In this proposal, as service terms are expressed as constraints over metrics in iAgree, the compliance operation is described with a Constraint Satisfaction Problem (CSP) using the metrics as variables, and the Terms as constraints. The statement "if and only if none of the offer Terms contradicts the creation constraints" is described with the CSP operation Satisfiability (i.e.: logic operator "exists") with the additional constraint: $CC\ AND\ !(ST)$, where CC is the set of Creation Constraints and ST is the set of constraints related to the Terms. With the evaluation of Satisfiability of this constraint, it is evaluated if exists a possible value for the metric that solves the creation constraints but does not solve the service Terms.

### 3.4.2 Agreement validity

Although some services are ruled by simple SLAs (such as SLA from Amazon EC where only Availability is guaranteed for values greater than 99.99% and 99.0%), in other scenarios, a single SLA can include a great number of guarantees with more complex SLOs (e.g.: in IT Maintenance, the average accomplishment of the committed times in the SLA have to be over 95%, where the committed times depend on factors such as the Severity of the maintenance or work force assigned). As SLAs are humanly

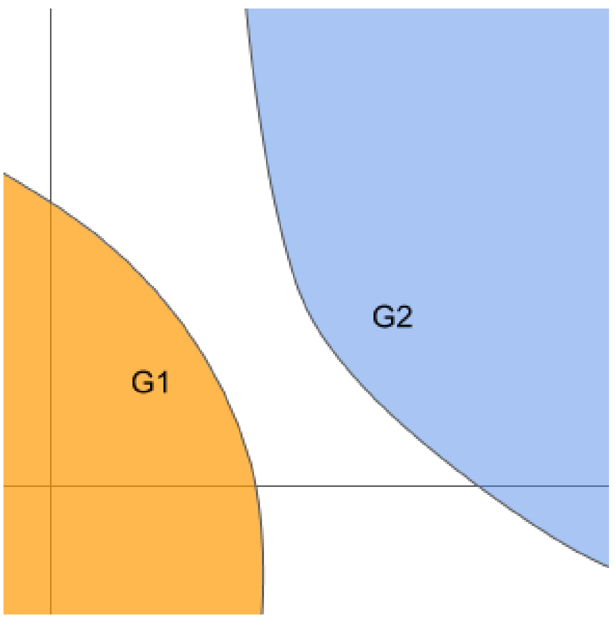


Figure 3.9: Inconsistent guarantees

created, even in simple cases the definition of an SLA can be a tedious and error-prone task. One complex analysis operation is the evaluation of the validity of the agreement. Some validity evaluations require a domain or semantic knowledge. For example, the guarantee:

$G1 : Availability < 80\%$

in common cloud computing scenario can be considered an error, as providers usually guarantee that the time when machines are available is *greater than* a percentage of total time, not the opposive, *lesser than* %.

In spite of domain criteria, other validity checking can be performed. This is the case of the agreement self-consistency. An agreement is considered consistent if there are no contradictions among their guarantee terms. For instance, these SLOs are contradictory:

$G1 : MUP > 99$
$G2 : MUP \leq 99$

This kind of inconsistency is depicted in Figure §3.9. We represent orange coloured the metric values which fulfill the SLO in G1 guarantee and blue coloured the metric values which fulfill the SLO in G2. As there is no overlapped areas for such values, G1 and G2 can never simultaneously be accomplished. Therefore, there was an error in the modelling of these guarantees or one of them is never going to be fulfilled.

Inconsistencies can even be isolated to single guarantees:

$G1 : MUP > 99\ AND\ MUP \leq 99$

The consistency operation takes the guarantees as input and returns a boolean regarding the consistency or inconsistency of the guarantees. Depending on the solving technique, we could consider to return the explaining of errors in case of inconsistencies.

Another evaluation to check the validity of the agreement is analysing if there are dead terms, i.e. terms that are never evaluable. A guarantee term is a dead term if it can never be applied. For example, when the scope of the guarantee can never be achieved (e.g.: a guarantee term which only applies when Availability < 0% and Availability is measured as a time percentage). Other more complex cases appear when the qualifying condition for a guarantee is inconsistent with other guarantee SLO:

$G1: \ ResponseTime < 2\ hours$
$G2: \ If\ ResponseTime > 2\ hours\ then\ SolvingTime < 1\ day$

In this example, G2 assumes that G1 can be violated. Without any information about what happens when an SLO is broken (i.e.: the SLA could be considered cancelled), this example can be either a model error either a model decision to apply different guarantees.

## 3.5 Tooling Support: ADA and Governify

Originally, the WS-Agreement specification has been successfully applied in the Grid computing arena within grids services in projects such as Assess Grid, a project funded by the European Commission [1], BEinGRID SLA negotiation component [50], UNICOREVIOLA, of CoreGRID and Viola [62], or decentralized, cross-middleware grid job submission service (JSS) [174], between others. Furthermore, The GRAAP Working group of the Open Grid Forum, lead by researchers of the Fraunhofer SCAI, developed WSAG4J, a framework developed with Java supporting the WS-Agreement protocol and considering all phases of the SLAs lifecycle. And it provides some operations as agreement offer validation based on template creations constraints or the evaluation of penalties and rewards for guarantees.

A more complete component stack is provided by Governify Platform, `http://www.governify.io`. This stack includes different components to provide a full ecosystem for the management of SLAs, including: (i) an editor with facilities to support the creation of valid SLAs, (ii) an SLA repository, (iii) an SLA monitor. In the Governify website, there is extensive information about SLAs modelling and analysis with a suite of the examples used in this work.

Governify is built on a microservice architecture basis so each service is developed and deployed independently and they are connected through REST APIs.

### 3.5.1 ADA: SLA Analysis

The analysis support in Governify is provided by ADA, which is a complete analysis tool that was developed in the context of Thesis from C. Müller [122] and delivered as a library for Agreement Documents' Analysis (ADA)[124][6]. The basis of ADA is defining each analysis operation as a Constraint Satisfaction Problem (CSP) and solve it (logic satisfiability operation). In an SLA, we find assignments to describe pricing or other service functional terms (e.g.: billing period: monthly), inequalities to describe SLOs (e.g. Availability > 99%) or piecewise functions to describe compensations function:

$Penalty = 10\%\ when\ 95\% \leq Availability < 99\%$
$Penalty = 30\%\ when\ Availability < 99\%$

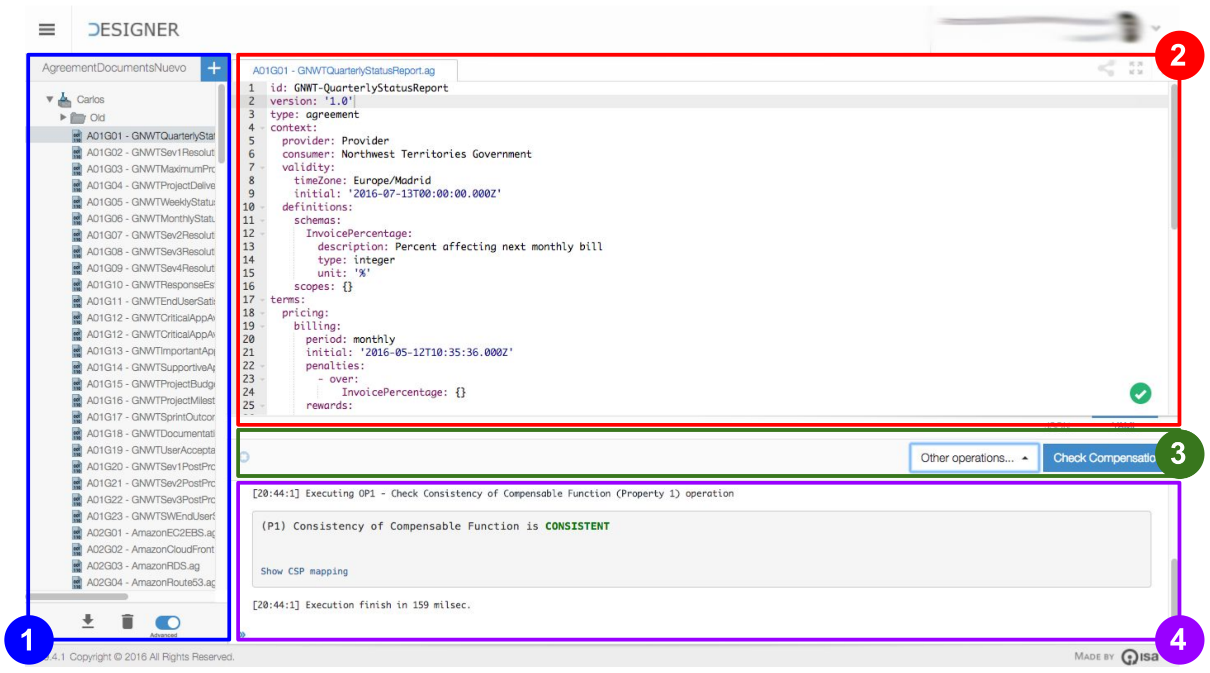
These sentences can be directly mapped to constraints. The ADA library provides the mapping to the different CSP related to each analysis operation. These CSPs are solved with Minizinc CSP. Each operation (including the mapping from iAgree to CSP) is exposed with a REST API.

### 3.5.2 SLA Designer

Governify provides an SLA editor, named SLA Designer, which internally uses iAgree with yaml syntax but it can be extended to other syntaxes such as json or xml. Figure §3.10 displays the main interface of SLA Designer. The rendering of the documents can be customized with document templates to speed up the creation of new SLAs just filling certain sections, such as target values in SLOs or even to enrich it with graphic visualization for compensation or metric functions. The use of iAgree enables to interact with ADA, the analysis library, to use their analysis operations to create valid SLAs in design time.

The editor window is divided in 4 sections. On the left side, section 1 in Figure §3.10, we find the project and file browser, where new documents can be created, organized or deleted. The documents are stored in the SLA Repository which is accessible with a REST API. In the main window, section 2 in Figure §3.10, the documents are edited. SLA Designer provides coloured syntax and other automatic features to speed up edition. The command panel, section 3 in Figure §3.10, provides the different analysis operations depending on the document (template, offers, SLAs). And, finally, the results of the operations are output into the console, in the section 4, where operations can be also manually invoked with autocompletion.

[6]http://www.isa.us.es/ada



```
id: GNWT-QuarterlyStatusReport
version: '1.0'
type: agreement
context:
  provider: Provider
  consumer: Northwest Territories Government
  validity:
    timeZone: Europe/Madrid
    initial: '2016-07-13T00:00:00.000Z'
  definitions:
    schemas:
      InvoicePercentage:
        description: Percent affecting next monthly bill
        type: integer
        unit: '%'
    scopes: {}
terms:
  pricing:
    billing:
      period: monthly
      initial: '2016-05-12T10:35:36.000Z'
      penalties:
        - over:
            InvoicePercentage: {}
      rewards:
```

```
[20:44:1] Executing OP1 - Check Consistency of Compensable Function (Property 1) operation

(P1) Consistency of Compensable Function is CONSISTENT

Show CSP mapping

[20:44:1] Execution finish in 159 milsec.
```

Figure 3.10: SLA Designer

### 3.5.3 SLA Repository

The SLAs are stored in a external component, the SLA Repository, which is exposed through an API to be integrated with the editor or the Monitor component. This API has been developed with Swagger Framework following the OpenAPI Specification to enable the development of the entire API lifecycle and can be completely consulted in `http://datastore.governify.io/docs/index.html`.

### 3.5.4 SLA Dashboard

Figure §3.11 depicts the main Dashboard of Governify, where the monitoring and evaluation of SLOs are graphically displayed. The Dashboard depicts the monitoring values which are consumed from the computers indicated for each metric in the SLA. A Computer is a specific object to process performance values (aggregating them, calculating averages, etc.) and have to be instantiated with parameters such as the URL where the related performance value can be queried (e.g.: pooling a REST API). The performance values are depicted together the KPIs values, the SLA status, etc.

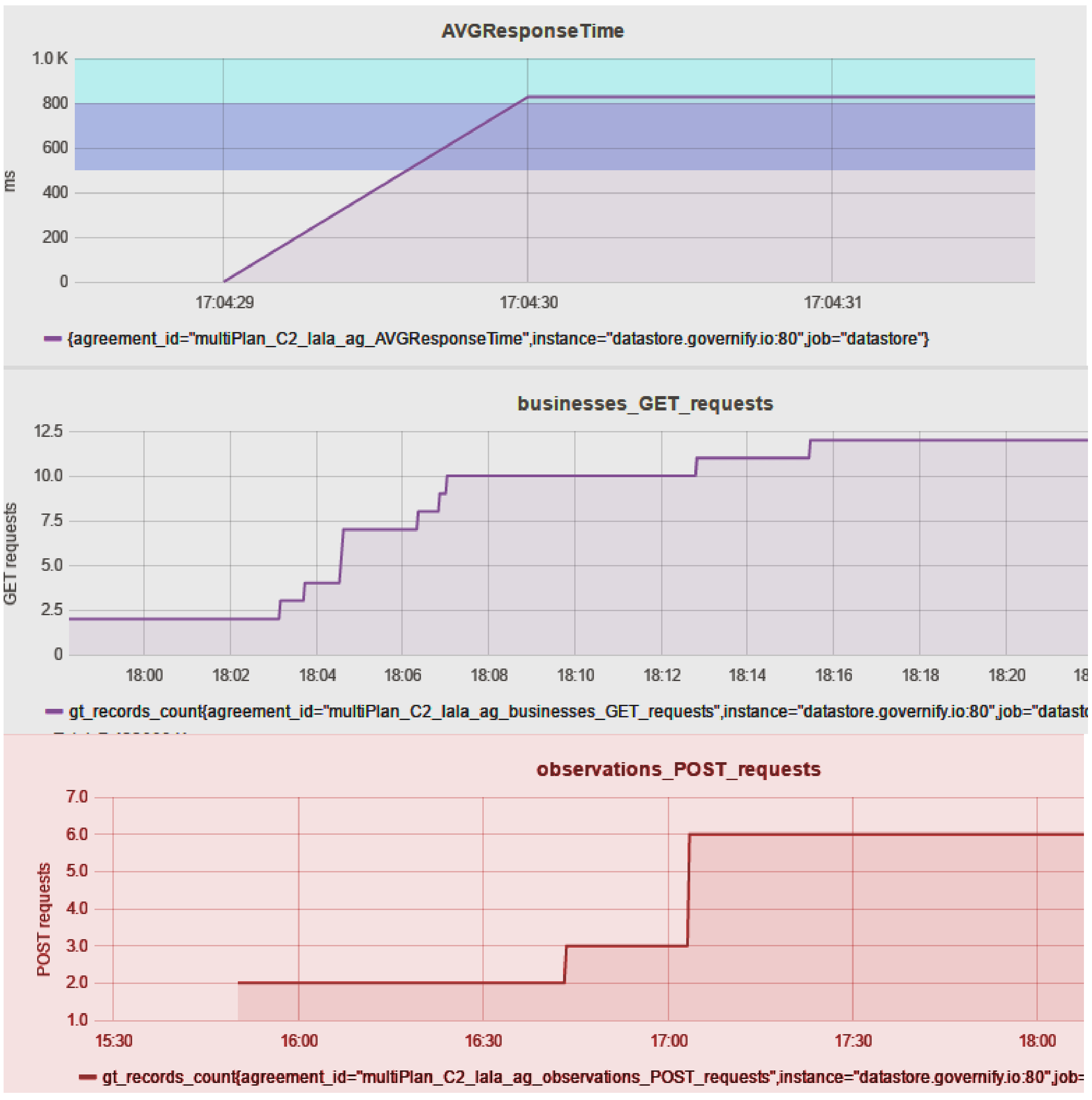


Figure 3.11: Guarantee terms monitoring with Governify

## 3.6 Other SLA Management proposals

We have developed an extense review of the research efforts related to the automated analysis of either namely SLAs or just quality of service (in the form of performance metrics) in different domains. The research sources were the common research databases, such as Scopus [7] or Google Scholar [8].

To introduce this analysis, we are going to distinguish three kind of works. First, the works which define a new SLA model including the service description together the parties guarantees, highlighting the domains where they were applied and the expressiveness of such languages. Then, we are going to analyse the works which focus on the definition of performance metrics based on the domain and their utility regarding SLA (in order to negotiate SLAs). These proposals either extend an existing SLA to describe specific domain metrics or just focus on the definition of the quality of service metrics (usually performance metrics) but without a complete SLA formalisation. And, lastly, we analyse the proposals related to use the evaluation of the SLA accomplishment to make decisions about service execution (scale resources, review execution plan,...).

### 3.6.1 SLA Models

The reviewed proposals in this subsection introduce or create a model to describe the main elements in an SLA: Service description, metrics and guarantees. In the analysis, we have identified the domain where the proposal have been applied: if is limited to a specific explicit quality vocabulary or it is service domain independent, the expressiveness of the SLA model (for example, if they include Templates or Compensations functions), and if they provide some tooling support for the SLA lifecycle.

SLAng [38] is a language, created in 2003, for agreement modeling proposed by University College London. This language intends to associate penalties with QoS properties defined in the agreement. In addition, it allows applying these restrictions dynamically at the time or by some state variable visible by all the parties to the agreement. They propose to fill the gap of QoS to sign SLAs in computational domains, as service descriptions are covered by other languages such as WSDL or business process definitions are covered by BPEL.

SLA* [93] is part of the European project SLA@SOI, which concluded in 2011, which sought to increase the automation and predictability of all phases of the agreement life cycle. The syntax is domain independent and describes agreement templates documents and agreement documents through a generalization of other specifications such as WS-Agreement, WSLA or WSDL. It is not especially focused on web services but allows it to be extended with the requirements of each domain. As in other proposals,

[7]http://www.scopus.com
[8]http://shcolar.google.com

an agreement template contains five sections: 1) template attributes; 2) the parties involved in the agreement; 3) service description; 4) variable declarations; 5) terms of the agreement and guarantees.

Service-Level-Agreement for Clouds (SLAC) [194] is a language, proposed in 2014, to define SLA in a cloud environment. Since it is WS-Agreement inspired, it shares many characteristics, structure and definitions. This language offers multiple predefined metrics, inspired by cloud computing, e.g. better multi-party support, group definitions, and parties involved for each term. Unlike other agreement models, SLAC does not differentiate between service description terms and QoS requirements.

rSLA [171] is a language, created in 2015, for specifying, monitoring and enforcing SLAs for cloud services. The language describes basic metrics that are to be obtained and how they are aggregated in terms of composite metrics. It is possible to define how to proceed if SLOs are met or violated. The syntax is similar to Ruby language (because of this the 'r' in 'rSLA'). As WSLA, they use conditions-actions to define the guarantees and they support renegotiation or compensations (actions after SLA violations). They support metric definition through composition and negotiation and monitoring of SLAs

Linked-USDL Agreement [70] is an extension to Linked-USDL [69]. It can capture agreement terms, business aspects, liability, compensations, and time constraints. Specifically, Linked-USDL Agreement is designed to be used to establish and share agreements among customers and providers that seek to perform automated service trading in a web context. They describe 2 examples: A business process outsourcing and a cloud service example. And provide tooling support for Document lifecycle including RDFa analysis (a subset of RDFservice).

Quality Requirements Language (QRL) [157] (published in 2002) is a fully fledged and expressive language to describe quality requirements of products. The quality requirements and the products can be considered as the SLOs and SDTs in the WS-Agreement specification, respectively. As WS-Agreement proposes the use of service properties and metrics to be used in the SLOs, QRL, similarly, describes a catalogue of attributes with their metrics to be used within the quality requirements.

Collaboration-Protocol Profile and Agreement Specification (CPPA) [135] (which was published in 2002) was proposed by OASIS based on an electronic commerce collaboration standard language called ebXML [137]. It is based on previous works developed in projects as Unified Business Agreements and Contracts (UBAC), performed in 2002 by the United Nations Centre for Trade and Facilitation and Electronic Business (UN/CEFACT). The main objective of ebXML is to obtain a setup document to be used at runtime called ebXML Business Service Interface Configuration. Thus, ebXML is a complete approach which wraps all necessary aspects to define the information exchange in SOA systems. The starting point to obtain the setup document is to perform the ebXML business process (ebXMLBP) [136], which is based on business processes standard languages as BPMN. Then, it is necessary to study the collaborations between the interested parties by means of different OASIS specification languages. For the business collaboration context they propose to use the Collaboration-Protocol Profile (CPP) specification. CPP describes the agreement party capabilities, including information about: each party, communication protocol, security, document exchange protocol, retries, and other information necessary for an autonomous interaction between parties. Once described the CPP, it is mandatory to define information system terms and constraints to allow the electronic exchange. Such terms and constraints are specified by the ebXML agreement specification called Collaboration-Protocol and Agreement (CPA). CPP and CPA are included in the Collaboration-Protocol Profile and Agreement Specification (CPPA) [135]. Since the CPPA specification is focused in electronic services, they provide only a limited set of domain-specific quality of service constraints on the kind of messages, certificates, protocols, etc. Therefore, the differences in the scope of CPPA and WS-Agreement specifications make negligible a comparison between them.

Rule-Based Service Level Agreement Language (RBSLA) [143] (published in 2005) is a declarative rule language which enhances the XML-serialised RuleML[9] language with useful constructs to express SLA terminology. RBSLA support to express predicates, event condition action rules, and rule priorities, that describes concepts similar to the SLOs, QCs, and relative importance of BVL, of the WS-Agreement specification. In addition, RBSLA supports specific vocabularies defined externally (e.g. in RDF-S/OWL ontologies[10], or Java class hierarchies) for elements such as metrics, pricing policies, rights and obligations. A great advantage is that off-the-shelf RuleML engines already exist for the processing of such SLAs in order to validate the SLA rules.

CC-Pi specification [20] (published in 2007) proposes a theoretical framework for mapping SLAs to service constraints. The CC-Pi model is, however, tightly coupled to the negotiation and monitoring process. This leads to the lack of concepts such as agreement parties or service interfaces. Nevertheless, CC-Pi supports to describe variables and constraints that connect the involved variables, as the metrics and SLOs of WS-Agreement specification. CC-Pi defines a validity criteria to reach and validate agreements using Constraint Satisfaction Problems (CSPs) that considers SLA requirements imposed by the involved parties.

A number of proposals have appeared with the rise of cloud services. At this regard, Stantchev *et al.* [168] propose to connect a set of guarantee terms defined with non-functional properties (metrics) and goals (SLOs) with specific performance issues in real cloud platforms. They are not very precise about how to handle the variability of platforms. From the formalisation of SLOs they propose to negotiate SLAs and enforce them when the connection for them is defined. Wieder *et al.* [194] define a *Service Oriented Architecture* with their own SLA model. The model has to be refined on each specific domain and there is an independent proposal to define measurements. Son *et al.* [167] introduce a full lifecycle SLA management for Cloud services with their

[9]`http://ruleml.org/`
[10]`http://www.w3.org/TR/rif-rdf-owl/`

quality own vocabulary (allocation region, time response, price...). The syntax of the SLA is a set of constraints with required performance levels (the common properties in cloud platforms). They use these offers to negotiate and, then, signing the agreement according to an utility function.

There are different proposals to model contracts for specific domains. This is the case of Daly [34] *et al.* , who propose a simple SLA language for CRMs. They describe metrics as customer satisfaction or utility, as the relationship between accomplishment and customer satisfaction. Augenstin *et al.* [7] propose a language to describe logistic services. They mainly focus on functional aspects. The performance quality is named, but none attribute or related language are proposed. They propose to model service descriptions from a BPMN using a model to model transformation. Nguyen *et al.* [133] propose align the SLAs for the different services/tasks with different layers: business, people and technology. So they describe different categories for SLOs and for SLA in the different layers and check how they align the other layers. Chau *et al.* [29] proposes to define performance metrics and goals based on business processes. they relates SLAs and business process artifacts where guarantees over the process are defined through process events.

Other proposals focus on different phases of the SLA lifecycle, so they define a model with a language with ease these phases. Therefore, Uriarte *et al.* [175], introduce a formal language based on CSP to describe SLA together different protocols for negotiation and monitoring. They highlight the advantage of using their language for service brokering and they provide a comparison between different generic SLA languages. Kotsokalis *et al.* [99] propose a syntax for describing guarantees in SLAs based on Binary Decision Diagrams (BDD) as the models proposed by WSLA or WS-Agreement do not include semantics for service description. They propose mapping qualifying conditions to conditions, and service terms as facts and guarantees as conditions. Dastjerdi *et al.* [35] proposed that the SLAs include semantic information so they are fully understood by all the parties. They also criticise the importance of the deployment time in Cloud services and, as the relationships between different layers (IaaS, SaaS, ..) are not described, they propose using Web Service Modeling Ontology (WSMO) for composing descriptions (between layers) and monitoring the deployment of service. Goo *et al.* [75] discuss about the role of SLA in service outsourcing and how to define metrics, and targets values. They also consider that the agreement metrics should not provided alone themselves, but also with the measurement procedure and what happens in case of violation (penalties or rewards). Grzech *et al.* [79] analyze the impact of performance on the service cost from a functional point of view. They view a service as a number of tasks that are described in the SLA, so the cost depends on how to manage these tasks. They model service composition as a more complex service (with more tasks) and they propose a model to analyse the execution order of the different tasks.

### 3.6.2 Specific domain SLA metrics

In this subsection, we analyse the different proposals related to the definition of a quality and performance vocabulary together with the SLA creation based on this vocabulary. We also highlight if they use an existing language they use to describe the metrics (as the ones in previous section) or use an ad-hoc language for the sake of their proposal.

Sauvé *et al.* [161] propose a methodology to calculate SLO thresholds to sign IT services SLAs according to service function cost from a business perspective. In all these cases, guarantees are proposed upon computational metrics (e.g. response time or availability). Therefore, it is useful only for SLAs that apply to the software infrastructure that support business processes and not for the business processes offered as a service.

Kieninger *et al.* [95] describe a categorization of IT services and outline a mechanism to obtain efficient SLOs for them. However, they do that in a conceptual level and do not detail how they can be formalised to enable their automated management.

Daly *et al.* [34] propose an SLA model based on the different elements in the service provision, *i.e.* application, servers, network, etc, related to service provision system.

Cardoso *et al.* [25] propose a description language for services that include business characteristics together with technical or operational parameters. This work is focused on managing services including business perspective. in this domain, there is a limited amount of work. The solution of Augenstein *et al.* [7] introduce a platform based on service-oriented- approach for managing contracts on 4PL business. The proposed solution itself is mainly focused on coordinating the business process conducted among these different partners.

Another example is the work introduced by Bing and Zhongying [15]. They define in mathematical terms the parameters of a contract in transport & logistics collaborative business process. Mai and Teo [83] also followed a mathematical approach to define and analyse contracts in the collaborative business process in transport & logistics. Nevertheless, none of the aforementioned solutions focus on the conformance check of the agreements among the partners.

In [78], Grubitzsch *et al.* focus on the definition of multiple objectives in Service Level Agreements to provide a management document for flexible services which adapt performance according to these multiple levels to avoid violations and cancelling of SLAs. They define different quality thresholds to enable SLA renegotiation or SLO update depending on the situation. They propose a pricing model for the different execution levels and penalties to take decisions about choosing provider.

In [133], Nguyen *et al.* propose to align the metrics guaranteed in the SLA to the actors related to the metric in manual processes (such as actors to satisfaction metrics or costs to used applications).

Bar-Isaac *et al.* in [10] 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, Fenghui Ren *et al.* analyse in [152] how utility function is obtained from customer objective function (i.e., customers timetable preferences affect how transactions distribute through commercial opening hours).

Correia *et al.* [32] relate SLA Lifecycle to BPMN lifecycle and they propose a table of relationships between SLA and BPM (e.g. A SLA Contract is related to a Process).

Terry *et al.* [172] propose an architecture to provide a storage service based on different storage providers according to a consistency metric. The consistency is based on data integrity, so storage services are databases. At the end, the proposed storage service selects the best service according an utility function based on this consistency evaluation together other common properties, such as latency or response time.

Again, there are also a number of proposals related to the cloud specific domain. Messina *et al.* [117] propose a multi-agent system to negotiate contracting over SLA based on performance metrics (CPU, Availability, Time to boot, response time, ...) and then propose a protocol for SLA selection with the agents. Omezzine *et al.* [140] describe requirements for SLA negotiation in cloud environments where there are different layers (IaaS, SaaS) with different terms categories to discuss. They propose an architecture and flexible procedure to calculate an utility measure to select the best resources in a Cloud. Wu *et al.* [196] proposes a broker for cloud services based on a cloud performance metrics (such as availability, response time, price or customer satisfaction level) and an algorithm to negotiate the optimal SLA dynamically. The SLOs are based on WSPolicy. Lam-Son *et al.* [103] propose decompose a service described by its high-level business goals to elementary constituent services so their elementary goals can be precisely defined and analyse as common basic utility properties (availability, inputs, outputs...) and then compose these properties. Zulkemine *et al.* [201] defines a protocol to negotiate SLAs based on the SLA goals. They differentiate between fixed terms, e.g. response time or free layer depending on VIP customers, and performance metrics with goals, like availability or price. The negotiation is performed by a Negotiation Broker and uses SLAs and a Knowledge Base to get an utility measure. Voorsluys *et al.* [187] introduce the cost of virtual machines live migration to the analysis of availability in order to relate the IaaS from the provider with the provided SLAs by the IaaS consumer to their SaaS customers. Yaqub *et al.* [198] propose a negotiation mechanism based on common quality metrics for cloud SLAs (availability, backup and performance) where different bidding strategies are applied to get the best deals.

### 3.6.3 Decision making based on SLA Analysis

In this section, we review the proposals which consider the evaluation of SLA to make decisions regarding service execution and, therefore, propose different strategies to enforce service based on the analysis of the accomplishment of the SLA (including the cancelling of the SLA in case of its violation). These proposals address how optimising the costs of resources planning or modelling complex scenarios to obtain an utility function to optimise the making decisions.

Hani *et al.* [86] review the different stages in SLA Management to (with related literature review) to conclude that renegotiation is better than cancelling SLA after SLO violation, so they propose different strategies for renegotiate SLA terms, based on theory's game or Bayesian learning. At this regard, Czajkowski *et al.* [33] provide a protocol to manage SLAs for resources planning. They relate the resources assignment with the negotiation mechanisms.

Tokairin *et al.* [173] propose use QoS and context information to optimize service bandwidth. They describe a protocol for presence and image detection to recognize when young/old people come into some place. Tan *et al.* [169] propose a mechanism to take decisions over business process execution based on its monitoring (in the context of Business Process Intelligence). They propose different metrics around abstract metrics and at different process perspectives (Cost, time, ...). Bobroff *et al.* [17] analyze a specific cloud scenario to allocate virtual machines in servers following a proposed strategy to ensure that SLA is not violated.

Buyya *et al.* [22] propose an architecture to allocate jobs from users in a Cloud environment considering the user QoS requirements (in the form of SLA) and infrastructure services (Amazon EC2 in the scenario). Buyya *et al.* [21] complement it with an allocation strategy for VMs based on SLAs from different providers and requirements as multi-objective function based on different metrics (price, performance..) and different providers (Amazon, Azure...). In a similar scenario, Y. Gao *et al.* [67] propose a model to save energy based on job predictions, SLAs (mainly response time requirements and possible violations), power costs, to manage resource in a data center environment. They introduce different mechanisms to avoid NP-hard planning cost.

Also in the cloud domain, Lorido-Botran *et al.* [110] make an analysis of different techniques to scale virtual machines based on different mechanisms. These techniques are basically based on action-reaction following different criteria, such as the SLA of Cloud services. And Garg *et al.* [72] propose a mechanism to distribute server load to maximize their use using SLAs documents from providers. The idea is preparing servers in advance without over provision. Linlin Wu *et al.* [195] propose an strategy to plan resources for new users in a SaaS environment. Their strategy considers two different SLA layers (SLA from user requirements/penalties and the SLA for resource provision) and a profit model based on the cost to execute the plan. Kailasam *et al.* [92] propose how to optimally manage Cloud resources considering public and private clouds (or different data centers), execution times of tasks, and transfer time. They use some quality measures, included in an SLA to propose an scheduler to make decisions about the processing of tasks in the cloud platform queue. X. Li *et al.* [108] provide a mechanism to monitor trust over the performance of cloud computing services, guided by SLAs. They also provide in [109] an architecture to manage multimedia personalized services based on QoS. They use an SLA broker who analyses customer requirements and provided QoS in SLAs (they use WS-Agreement) to choose the best platform and monitor it.

Emeakaroha *et al.* [58] propose an architecture to measure metrics and detect SLA violations in scenarios related to computing services. They define abstract metrics related to performance metrics: CPU availability, RAM and Storage, and different target values. They map performance abstract measures that are guaranteed in the SLA, like Availability, to specific monitoring variables, like Server Downtime in order to measure them without ambiguity. Similarly, Brandic *et al.* [19] propose a mapping based on WS-Agreement and XSLT. from abstract SLA metrics to the real taken measures.

Hedwiq *et al.* [88] propose a model for dynamic SLA management. They consider that the SLA is important for planning and providing resources based on the pricing and penalty model and it is required to adapt to the service using (the example are related to evolution of Wikipedia workload) so it is a kind of renegotiation of SLA as they propose to update SLO targets in runtime. They also relate this proposal to the changing pricing model in Amazon computing units.

Leitner *et al.* [104] propose monitoring SLAs in runtime to predict SLA violation in simple and composed services.

## 3.7 Summary

In this chapter we have introduced the SLA structure according to WS-Agreement and iAgree, as extension to provide a more human-readable language. We also detailed the SLA phases in which SLAs may be during its lifecycle with the aim of describing how these phases are considered in the agreement creation and their status proposed by WS-Agreement. We have provided the most common scenarios of the WS-Agreement protocol about SLA lifecycle. In turn, regarding the management of this lifecycle we have cited the most common operations to support it. Specifically, we have described the different states that agreements and guarantees may have during the WS-Agreement protocol. Such states ease the evaluation of agreements and guarantees at monitoring in order to detect violations. However, domain-specific monitoring techniques are not provided by WS-Agreement specification due to its domain-independent purpose. Such a domain-independence of the specification boosts the appearance of a plethora of tools with a diverse support for the creation and monitoring of WS-Agreement documents. At the end of the chapter we have provided information about most important of these tooling support approaches. Furthermore, we have detailed an extensible framework, Governify, which provides an enriched-text editor for the creation of iAgree documents with edition templates and a dashboard to monitor external systems based on the performance measuring and the SLAs. Lastly, we provide an extensive review of works related to the automated analysis and evaluation of SLAs. As we discussed, SLA management has been extensively researched in the computational domain. A diversity of languages and tools have been developed. However, in non-computational domains domains, SLA management is just beginning to be applied. The SLA management in other domains involve the increasing digitization of SLA documents together with the need to properly manage the information in the SLA to enforce business services.

# Part III

# Our Contributions

# SLA Model for BPs

*Everything I do and everything Pixar does is based on a simple rule: Quality is the best business plan, period.*

*John Lasseter (1957–),*
*2009*

## 4.1 Introduction

As we have introduced, non-computational services are process-oriented and the software that supports them is usually a process-aware information system (PAIS) such as ERPs, Service Desk Management Systems, CRMs, or business process management systems (BPMSs). However, unlike computational services, there is little work related to the extension of PAISs with SLA-aware capabilities to support non-computational services. A PAIS with SLA-aware capabilities, i.e. an SLA-aware PAIS, is a PAIS that uses explicit definitions of SLAs to enable or improve the automation of certain tasks related to both the SLAs and their fulfillment such as performance monitoring, human resource assignment or process configuration [179].

In this chapter, we formalise the SLAs for BP as a first step to enable such SLA-aware PAIS. To this end, after analysing the modelling requirements of such SLAs, four main aspects involved in their formalisation have been identified, namely: 1) the description of the business process provided by the service; 2) the SLOs guaranteed by the SLA; 3) the penalties and rewards that apply if the guarantees are not fulfilled; and 4) the definition of the metrics used in these guarantees. These SLAs are usually described in natural language so we provide a formalisation for aforementioned aspects using computational SLAs and techniques used to model the service description together with the process performance indicators and goals. Furthermore, we validate our approach through the modelling of several real BP SLAs.

The contribution of this chapter regarding the scope of this dissertation is highlighted in Figure §4.1.

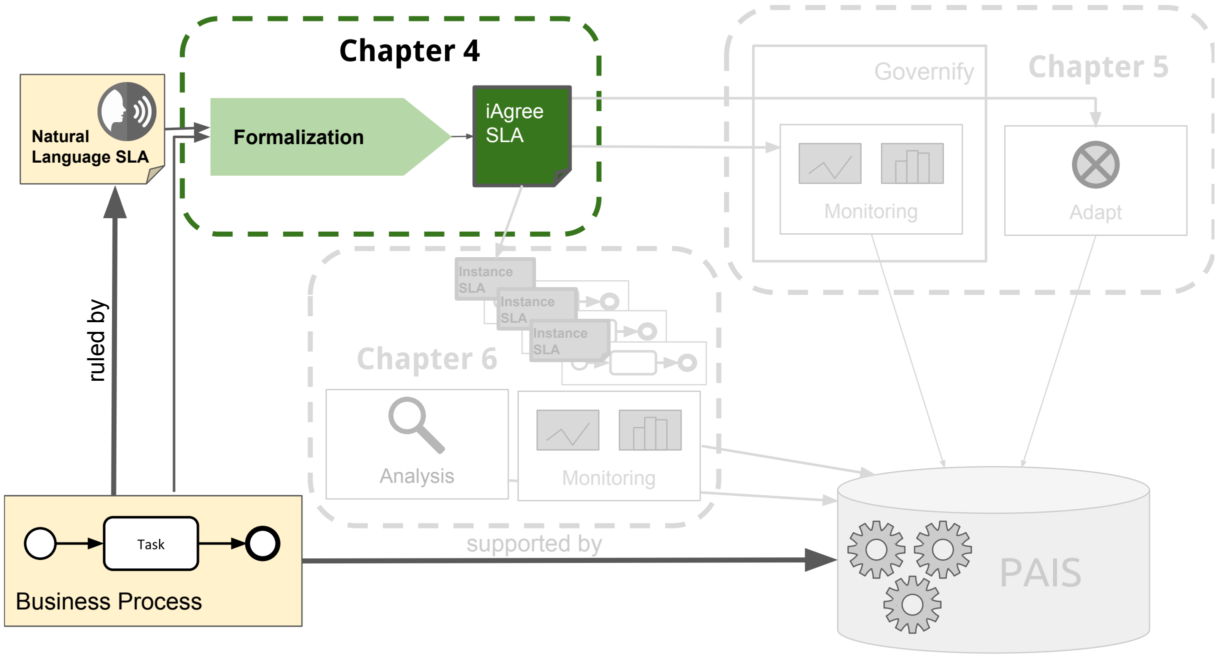


Figure 4.1: Contributions background

## 4.2 An example scenario

IT maintenance tasks are commonly outsourced agreeing to an SLA. We take as an example of these SLAs the definition of statements of technical requirements (STR) of a public company which belongs to the Andalusian Autonomous Government, from now on APC, acronym of Andalusian Public Company. Statements of technical requirements are described in natural language and include information about the services required as well as their SLA. The statement of technical requirements document of this example is defined for the Technical Field Support for the Deployment of the Corporative Telecommunication Network of the Andalusian Autonomous Government. It is presented in a 72-page document written in natural language including the SLAs defined for five of the required services, namely: 1) field interventions; 2) incidents; 3) network maintenance; 4) installations and wiring; and 5) logistics. In particular, we focus on the field interventions (FI) service. The term *field intervention* references to the fact of requiring the presence of a technician at any headquarter of the APC for different reasons: troubleshooting technical assistance, installations supervision or restructure. From a high-level perspective, the FI service is performed as follows: the APC requires an FI, which can have different levels of severity, from the contractor staff. Then, the contractor plans the FI and performs it at the infrastructure location. In some cases, the contractor must provide some required documentation and, if such documentation is considered incomplete or inadequate by the APC, it needs to be resubmitted by the contractor until it fulfills the APC's quality requirements.

For this service, the statement of technical requirements (STR) document presents the following information: 1) the committed times by the contractor (see Table §4.1); 2) the general objective defined for FIs —the SLO of the SLA— represented as AFIP $>$ 95%, where the AFIP (*accomplished FIs percentage*) metric is defined as:

$$\text{AFIP} = \frac{\text{\# accomplished FIs}}{\text{\# FIs}} \times 100$$

and 3) the penalties applied in case of the SLO is not accomplished (see Table §4.2). These penalties are defined as a percentage over the monthly billing by the contractor for the FI service. In addition, the STR document describes the following definitions for the referred times in Table §4.1:

**Response Time:** Elapsed time between the notification of the FI request to the contractor and its planning, including resources assignment, i.e. technicians.

**Presence Time:** Elapsed time between resource (technician) assignment and the beginning of the FI, i.e. technician arrival.

**Resolution Time:** Elapsed time between the technician arrival and the end and closure of the FI.

**Documentation Time:** If documentation, i.e. reports, is required, it is defined as the elapsed time between the end and closure of the FI and documentation submission. If the APC considers this documentation as incomplete or inadequate, it will be rejected and sent back to the contractor and documentation time is again activated and computed.

The different stages of the service during the process are supported by different tools. These tools are not BPMSs but Process Aware Information Systems, where the different events in the service are tracked and orchestrated with the users interaction.

| **Criticality Level** | **Response Time** | **Presence Time** | **Resolution Time** | **Document Time** | **Timetable** | **Calendar** |
|---|---|---|---|---|---|---|
| Critical | 0.5 | 4 | 2 | 4 | 8:00 – 20:00 | Local |
| High | 2 | 8 | 4 | 12 | 8:00 – 20:00 | Local |
| Mild | 5 | 30 | 6 | 24 | 8:00 – 20:00 | Local |
| Low | 5 | 60 | 8 | 48 | 8:00 – 20:00 | Local |

Table 4.1: Committed times by the contractor (in hours) for the FI Service SLA

| AFIP | Penalty |
|---|---|
| 94% $\leq$ AFIP $<$ 95% | -1% |
| 93% $\leq$ AFIP $<$ 94% | -2% |
| 92% $\leq$ AFIP $<$ 93% | -3% |
| 91% $\leq$ AFIP $<$ 92% | -4% |
| 90% $\leq$ AFIP $<$ 91% | -5% |
| AFIP $<$ 90% | -10% |

Table 4.2: Penalties definition (in monthly billing percentage) for the FI Service SLA

## 4.3 Requirements for Modelling SLAs of BPs

Based on the analysis of the state of the art in SLAs for both computational and non computational services, and more than 20 different BP SLAs developed by 4 different organizations, we conclude that four elements must be formalised in SLAs for non-computational services, namely: 1) the business processes; 2) the metrics used in the SLA; 3) the SLOs guaranteed by the SLA and how to evaluate each SLO by filtering different parameters (e.g. region); and 4) the penalties and rewards that apply if guarantees are not fulfilled or over fulfilled. Next we describe each of them.

### 4.3.1 Business Processes

An SLA is always related to one or more specific services. The way such services must be provided is usually defined by describing the underpinning business process, and this is often done in natural language. Consequently, the formalisation of SLAs for non computational services requires the formalisation of the business process itself. Note that it is not required for the SLA to detail the low level business process that will be enacted by the provider's PAIS since most SLAs do not delve into that level of detail and just focus on main activities and the consumer-provider interaction (cf. Fig §4.2 for the high-level business process of the motivation scenario). The Field Intervention process follows a simple task sequence: Plan the Field Intervention, Perform and Document it (in the case it is required). The customer reviews the documentation submitted and can require corrections so the last task would be repeated. The process description can be as simple as a single sequence of tasks where tasks are executed and the deliverable of one or more tasks is evaluated to decide wether to continue with the following task or go back to execute a previous task in the sequence. Therefore, the service could be described with a generic language such as BPMN or in other cases with state machines, Event Process Chains, etc.

Task sequences are currently supported by a number of PAISs without requiring a generic BPMS. There are PAISs which support well-known and configurable workflows, such as Issue & Project Tracking System Jira or systems which support generic pipelines for project management, such as Trello and Liquid Planner.

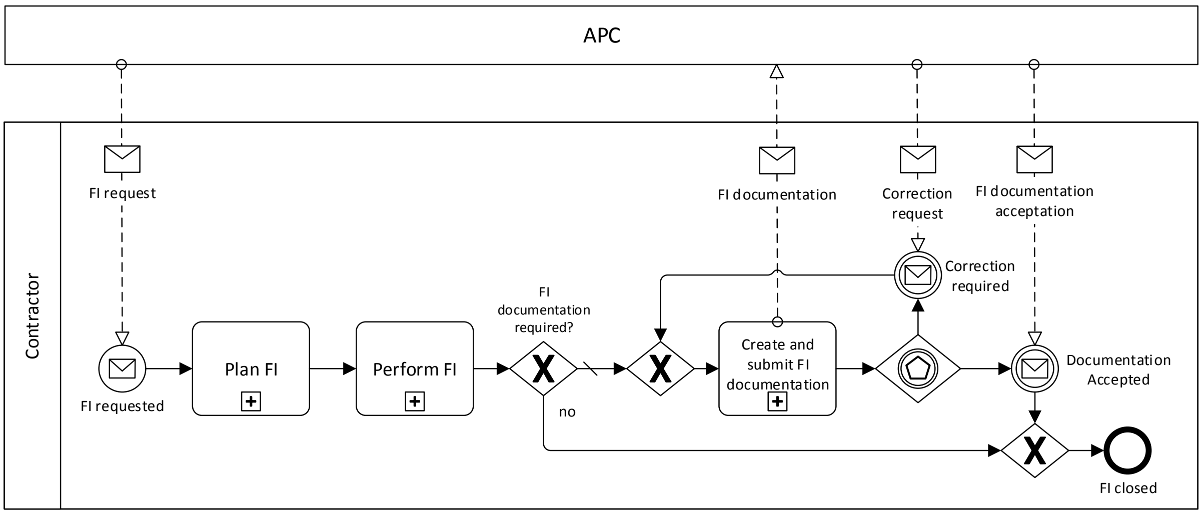


Figure 4.2: BPMN model of Field Intervention (FI) service

## 4.3.2 SLA metrics

The SLA metrics are measures that need to be computed so the fulfillment of the SLA can be evaluated. For instance, in the running example, the response time, the presence time, or the AFIP are examples of such metrics. The mechanism used to define these metrics must have three main features. On the one hand, it must be expressive, i.e. it must allow the definition of a wide variety of metrics. On the other hand, it must be traceable with the business process so that it enables their automated computation. And, lastly and more important, it has to be computable. In addition, it is convenient that the metrics are defined in a declarative way (i.e. defining what is measured instead of how to measure it) because it reduces the gap between the SLA defined in natural language and the formalised SLA and decouples the definition of the metric from its computation [47].

## 4.3.3 Guaranteed SLO

SLAs include assertions, namely Guarantee Terms, over the aforementioned metrics that are guaranteed by the SLA and, hence, must be fulfilled during the execution of the service. For instance, the running example defines AFIP $> 95\%$ as an Service Level Objective (SLO) for AFIP metric of the FI service. In general, SLOs are guarantees defined as mathematical constraints over one or more SLA metrics. The fulfillment of these constraints can be delimited to a specific time period which also can match the billing period. In the introduced example, the guarantee $AFIP \geq 95\%$ refers to the monthly billing period. Therefore, when the AFIP is less than 95 just for a week, the SLO is not considered unfulfilled (although preemptive management could consider this situation as potentially risky). Furthermore, the SLO can be evaluated as a whole or it is can be evaluated regarding a more specific scope. For example, if the APC has offices in different regions and requires to monitor the fulfillment of the AFIP SLO for each region.

### 4.3.4 Compensations

Guarantee terms usually include the compensations that are applied when the SLO is not fulfilled or is improved, respectively. An example is shown in Table §4.2, which depicts the penalties that apply for the FI Service SLA in our running example. The specification of penalties and rewards requires the definition of a mathematical function, whose domain is one or more SLA metrics and whose range is a number which represents the penalty or reward. The compensations are usually defined as a percentage over the billing. This percentage can be applied to the billing period when the SLO was not accomplished or to a future billing (e.g.: when the compensation is claimed). In the example scenario, when the expected times are not average fulfilled over a 95%, the contractor is penalized with an increasing reduction of the billing up to a 10% over the monthly billing.

## 4.4 SLAs for BP Services with iAgree

From the identified requirements, we conclude that the structure of SLAs for non-computational services can follow a similar structure to the SLAs defined for computational services. For instance, Amazon EC2 SLA[1] also includes a definition of the service; some metrics like the *monthly uptime percentage* (MUP); a guarantee defined as MUP $\geq$ 99.95% with a penalty based on the MUP and defined in terms of a percentage over the price paid in the last month. Therefore, the definition of guarantees is similar for computational and non-computational services.

In contrast, the description of the service and the definition of the metrics of non computational and computational SLAs presents significant differences. The main reason is that, unlike computational services, non computational services are process–aware and, hence, their description and their SLA metrics are based on that process.

Based on the similarities and differences between non computational SLAs and computational SLAs, we propose modelling non computational SLAs by combining the agreement structure and mechanisms for the definition of SLOs, penalties, and rewards that have been already proposed for computational SLAs, such as iAgree [122, 125] and notations used to model the processes and *Process Performance Indicators* (PPIs), such as [41, 49, 145, 149, 193], which were introduced in Chapter §2.

In the following we describe the specific information added in the different sections in the *iAgree Configuration* to support the business process domain. iAgree Configura-

[1] http://aws.amazon.com/ec2/sla/

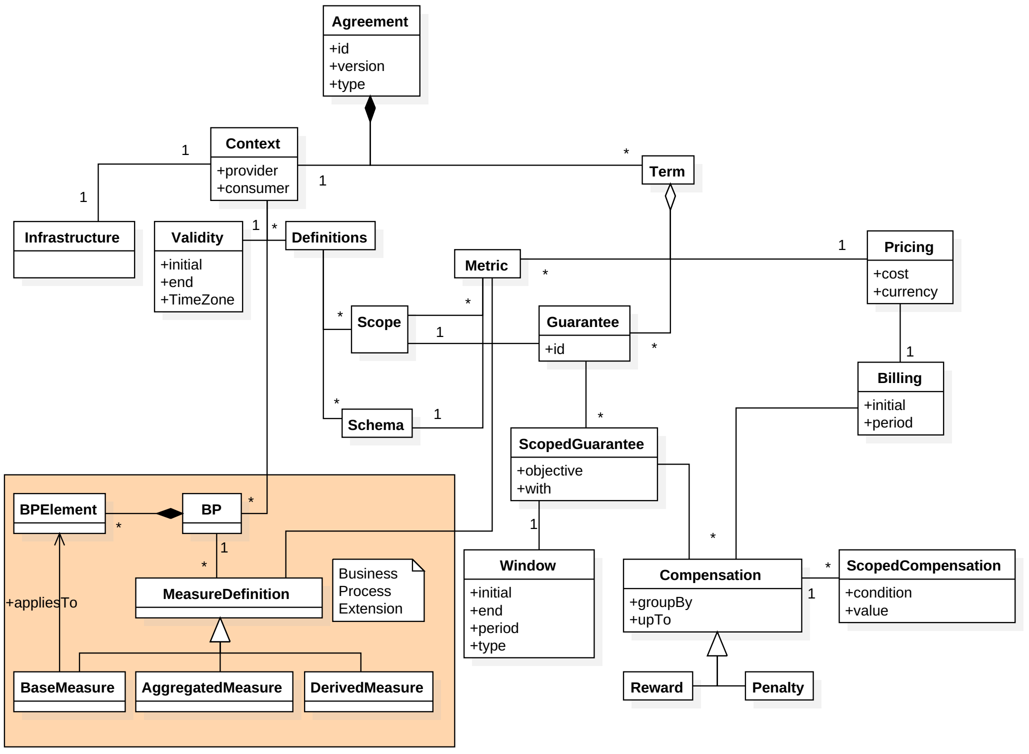


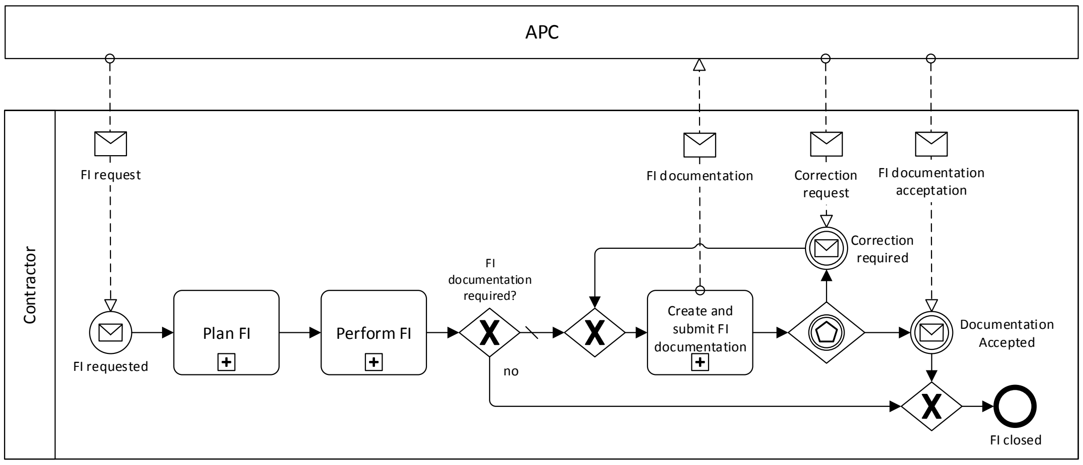
Figure 4.3: Class Diagram for Modelling SLA of a BP service

tion was described in Section §3.2.3 as the specification of the languages used in each section in the SLA structure. Specifically, we extend iAgree [126, 127], to support the requirements identified for our scenario, i.e. define the provided business process and its related metrics and guarantees. For such a goal, we propose BPMN as the language to model business processes and PPINOT [41] as the mechanism to model PPIs. These proposals have been chosen because of two reasons. Firstly, they are amongst the most expressive proposals of their kind, which is necessary to model the different scenarios that appear in BP SLAs. Secondly, they have a formal foundation that enables the development of advanced tooling support that can be reused in SLA–aware PAIS environments: (i) Governify, which was introduced in section §3.5 and (ii) *PPINOT Tool Suite* [42], which includes the definition of PPIs using either a graphical or a template–based textual notation [44], their automated analysis at design–time, and their automated computation from event logs. The class diagram for such extension is depicted in Figure §4.3, which extends the diagram §3.2 to support this scenario.

To follow the extensions we are going to use the running example, which is depicted in Figure §4.4 and fully described in appendix §C.2 but we illustrate also separately each relevant fragment of the SLA.

### 4.4.1 iAgree extension

```
id: FI_Service_SLA
version: '1.0'
type: agreement
context:
   process:
```



```
  definitions:
    schemas:
      MonthlyFeePercentage:
        description: Percentage affecting next monthly bill
        type: integer
        unit: '%'
    scopes: {}
  computers:
      responsetime:
        url: 'http://ppinot.computer.chap.governify.io:8080'
        apiVersion: '6'
terms:
  metrics:
    AFIP:
      schema:
        description: Average Field Interventions finished on time
  computer :
    $ref : '#/context/definitions/computers/AFIP'
    type : consumption
  guarantees:
    - id: G1
      of:
        - objective: AFIP > 95%
          with:
            window:
              type: static
              period: monthly
            penalties:
              - over:
                MonthlyFeePercentage
              of:
                - value: '95 - AFIP'
                  condition: 90% <= AFIP < 95%
                - value: '10'
                  condition: AFIP < 90\%
                 ...
```

Figure 4.4: Excerpt of the FI service SLA in *iAgree* syntax

### Context

The service description can be provided in terms of the underpinning business process and included in the *context* section. This description could be formalised with a generic process notation, such as EPC, finite-state machine or BPMN (*Business Process Model and Notation*), as in the example service (lines 5–11 in Figure §4.4). BPMN is a well-known standard to model Business Process with extensive tooling support. We depicted the abstract BP without considering execution information, which is usually BPMS dependent, for exemplification criteria, but it is not limited to it. It is common that the public BP is a business view of the deployed BP but the level of detail that consumer and provider agree. For example, specific endpoints for message communication could be included or the role assignment to user tasks. As BPMN can also address these implementation details, we base on it for the complete service description.

### Metrics

In non computational services, these metrics can be specified using a PPI–oriented approach. Specifically, metrics are defined using PPINOT measure definitions (described in Section §2.3). The PPINOT measure is defined as a metric computer with provides an endpoint that is referred in the SLA, so the metric becomes independent of the details of its processing.

Therefore, in the SLA document we find the metric name with the reference:

```
AFIP:
      schema:
        description: Average Field Interventions finished on time
  computer:
    $ref: '#/context/definitions/computers/AFIP'
    type: consumption
```

where the attribute *type* indicates that we have to make requests by pooling to the endpoint indicated by *$ref* attribute in the schema, which refers to the computer attribute in the context, to get the performance values.

And this url points to a document with the full PPINOT measure definition for the SLA (which is included in the appendix §C.2.1). Templates to describe PPIs are proposed in [44] in order to structure the information in a fixed form, reducing ambiguity, promoting reuse and also serves as a guide to avoid missing relevant information. According to these templates the PPI description can be described as:

```
AFIP:   The PPI value is calculated as the function
                  AFI_Measure/FI_Measure x 100,
                  where AFI_Measure is the sum of accomplished FIs
                  and FI_Measure is the number of FIs
```

PPINOT Derived measure refers an math or logic operation over other measure/s. In this case, the number of accomplished FI on time divided by the total number of FIs as a percentage. This metric is included in the only guarantee term (line 20 in Figure §4.4).

### Guarantee Terms

To define SLOs, we use the predicate language defined in iAgree [126], which includes relational, logical and common arithmetic operators. As apart from a concrete syntax, iAgree also provides semantics to define SLOs expressions as logic constraints, it enables the automation of analysis operations on SLAs such as detecting conflicts within an agreement document [126] or explaining SLA violations at run–time [129]. Furthermore, the scope attribute can be used to parameterise guarantee and their evaluation. For example, in the IT maintenance scenario, the guarantees for User Support are commonly scoped to three variables: *region*, *node* and *priority*. The service provided covers a wide geographical area, so the customer organizes the service in 8 different main *regions*. As these 8 main *regions* are known and statically defined, 8 different guarantees could be explicitly defined, but using a scope variable as facility enable us to define the SLO only once. *Nodes* are small offices which can be created during the SLA validity period so if we define a single different guarantee for each *node* in the SLA, the SLA would have to be updated each time a new *node* is created. When agreed guarantees can be applied to new *nodes* (for example under certain limitations such as new node within a city limits), the scope variable enables to define a single SLO for all the *nodes* without including the explicit domain for the *node* variable or listing exhaustively all the *nodes*. And, lastly, guarantees for this contractor are also scoped by *priority*. In this case, the SLOs for different Priorities are different (i.e.: Committed Response Time for Critical priority issues is lesser than for Normal priority issues). Therefore, the definition of a guarantee has different SLOs for different *priority* values although other commons attributes in the guarantee can be used, as the other scope variables or compensations.

In the running example, there is only one guarantee, regarding the previous defined metric AFIP (and line 29 in Figure §4.4):

```
- id: G1
    scope:
        {region, node, priority}
    of:
      - objective: AFIP > 95%
        window:
            type: static
            period: monthly
...
```

This percentage is measured according to a monthly static window, i.e. natural month. Time windows were described in section §3.2.2 as static (e.g.: natural weeks) or dynamic (e.g: the next or previous 7 days from a specific day). Although the committed times for each FI depends on the severity, the guarantee expects a global 95% of accomplished times, without considering which was the specific committed time for each FI (95% of total FI must accomplish their committed time).

### Compensations

Concerning penalties and rewards, they are defined with iAgree syntax as well together with the notion of *compensation function* defined in [127] and referred in Section §3.2.2.

Regarding the existing modelling proposed in section §3.2.2, there are no significant extensions to the compensations definitions.

In our example, as depicted in Table §5.2, when the AFIP is under 95% a penalization value is applied. This value goes from 1% to 5% percent when the AFIP is from 94% to 90%, respectively (reverse proportion) and a 10% is applied when the AFIP is under 90%. This percentage is regarding to the Monthly Fee, which is consistent with the monitoring period:

```
Guarantees:
   - id: G1
   ...
   with:
   ...
       penalties:
         - over:
           MonthlyFeePercentage
         of:
           - value: '95 - AFIP'
             condition: 90% ≤ AFIP < 95%
           - value: '10'
             condition: AFIP < 90%
```

## 4.4.2 Modelling the SLA

The SLAs are usually completely described in natural language. We describe here some considerations to formalise the natural language description to iAgree. First, some legals aspects, such as exclusion situations because natural disasters, are not covered by our formalisation and are out of the scope of this dissertation. Then, in order to model the SLA we differentiate among the service description, the service metrics and the guarantees.

### Service description

We base our proposal in describing the service with BPs. However, sometimes the natural language description is ambiguous so there is no a precise formalisation into business processes and it requires from a business expert to support this modelling. Furthermore, this description is usually an abstract modelling without execution details for the sake of understanding the service. Therefore, in some scenarios is even possible not modelling the full process, just the relevant events for the performance monitoring of the SLA. Performance metrics are addresses in next subsection.

#### Metrics

The formalisation of performance metrics into PPI metrics also requires a domain expert. This formalisation is based on the business process modelling. In order to describe the PPIs, it is only require to properly identify the significant events for the metric. For example, in the Field Intervention scenario, the Response Time metric requires identify the corresponding starting and finishing event to measure this time. The events, tasks or decisions that are executed between these two events are not relevant for describing the metric. In fact, as we introduced, it is usual that the business process described in the SLA is abstract so it is not executed. However, a precise correlation between the described metric and the real executed events is required in order to properly measure it. For enable, this would enable to perform monitoring from an event log as we describe in next chapter §5.

#### Guarantees

SLOs, as a fundamental part of the SLA, are usually straight forward defined. For example, "the availability of the service aims to be over 99%". However, in some cases, this goal has no compensation functions definition in case of underfulfillment which hinders how to manage the violation of the SLO. In other cases, the SLO is not directly described, just the compensation definitions. In these last cases, following the discussion in appendix §A, the SLO can be defined using the limit of the compensation region (e.g.: if it is defined a penalty for availability under 99%, we define the SLO as "availability >= 99%").

## 4.5 Applicability of our approach

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

- **RQ1: How expressive is our SLA language in comparison to real-world services?**. We want to know whether our service model is expressive enough to describe a wide variety of real-world services and which are the characteristics of the services that we are not able to express.

- **RQ2: Are we able to model the metrics in real-world SLAs with our SLA language?** Real-world metrics are expressed in natural language with complex considerations. Therefore, we examine the exceptions that may appear to measure them.

- **RQ3: Which difficulties appear when modelling SLAs?** SLAs are expressed in natural language with different parameters. The analysis of the SLAs is a fundamental goal of this dissertation, so we want to know whether real-world SLAs can be modelled and the problems that may appear to fully support them.

| Scenario | Service | Metric | Des. | Sc. | Excluded |
|---|---|---|---|---|---|
| 1. IT Systems by APC | User Support | 9 | ✓ | 3 | Calendar & work hours |
| | Gehronte | 10 | ✓ | 1 | Calendar & work hours |
| 2. Telecomm. Network by APC | Field Intervention | 1 | ✓ | 0 | Calendar & work hours |
| | Incidents | 1 | ✓ | 0 | Calendar & work hours |
| | Net. Maintenance | 1 | ✓ | 0 | Calendar & work hours |
| | Installations | 1 | ✓ | 0 | Calendar & work hours |
| | Logistics Service | 2 | ✓ | 0 | Calendar & work hours |
| 3. Project Management by APC | Performing Tasks | 4 | ✓ | 0 | Time offsets |
| | Staff Replacement | 3 | ✓ | 0 | Time offsets |
| | Availability | 1 | ✓ | 0 | Human resources |
| 4. Systems in Northwest Territories | Reporting | 3 | ✓ | 0 | ✓ |
| | User Support | 8 | ✓ | 1 | Calendar & work hours |
| | App Enhancement | 8 | ✓ | 1 | ✓ |

Table 4.3: Analysed Scenarios

In order to validate the applicability of our approach, we have used it to model the SLAs of 13 different services designed by 4 different organisations, with a total of 52 metrics and related guarantee terms. All these SLAs were published on Internet. The modelled SLAs are available at `https://github.com/isa-group/sla4bpdatasets`. In the case of the first scenario for IT Systems, we have not only modelled it but applied in the context of two transfer of technology projects, collaborating with private companies. For that reason, detailed real data about that scenario can not be provided.

The comparison table of this analysis is depicted in Table §4.3. The first column indicates the scenario modelled where APC indicates an Andalusian Public Company which acts as the contractor. The second column, Service, names the different services in the scenario. The third column indicates the number of KPIs defined for each service. The fourth column, Des., indicates if the Service DEScription could be completely described with BPMN. The fifth column, Sc., indicates the number of scopes defined in the scenario. And the sixth column, Excluded, indicates if the metrics described in the Guarantees could be completely described with PPINOT or there were limitations to be further handled, which are expressed in the cell.

### 4.5.1 RQ1: Service Expressiveness

We propose using a formal BP language, such as BPMN, for the service description together PPINOT for the metrics and iAgree for the general SLA structure together with the guarantees. As BPMN provides enough expressiveness to model process semantics involving task flows and orchestrations and user or computational tasks, all the analysed services could be described with BPMN in the context section. As we introduced in the Section §4.3 common maintenance scenarios follow a single task sequence with simple control flows as executing one of the tasks in the sequence or not, or go back to a previous task. The only issue is that the service is usually described with natural language, so the formalisation of the SLA requires that a domain expert supports the definition of the BPs without ambiguity.

### 4.5.2 RQ2: Metrics

Concerning SLA metrics, although most of them could be successfully modelled using PPINOT, there were a few types that could not be represented properly. As far as we know, this limitation is not specific to PPINOT, since there is not any other PPI modelling approach that can model all of the metrics that appear in the analysed SLAs.

We believe that the main reason why we have found this limitation is that, although related, the purpose of PPIs and SLA metrics are slightly different. PPIs are used internally by the organisation that performs the process as a mechanism to improve its performance. In contrast, SLA metrics are aimed at providing service–level guarantees to the service consumer or defining penalties when guarantees are not met. As a consequence, SLA metrics are much more focused on the customer and its expectations than the former.

Previous to this dissertation, parameters such as work calendars, were not supported by PPINOT but it was extended because they appear in all the SLAs analysed.

Specifically, we found three types of metrics that cannot be modelled neither with PPINOT nor with most of the other PPI modelling approaches:

**Metrics which involve complex date definitions** It is common that in order to measure dates and times, we have to consider idle times, such as office working hours, or holidays (or just weekends) so the measure is correct. In other cases, it is also common taking an Event trigger as reference (i.e. the start of a Task) but use some specific delay or advance for that Event (i.e. the duration between "2 hours before a Task starts" and "2 hours after it finishes").

**Metrics that involve human resources** These metrics are used in SLAs in which the task performer profile must be taken into account when applying penalties, so that a different coefficient is applied, according to the different profiles, to calculate the penalty. For instance, in one of the studied scenarios, the general penalty of each metric had to be multiplied by the monthly profile rate of the person involved in the non-fulfillment. This metric is again closely related with the customer. In this case, with the fact that the customer expects a fair compensation depending on the task performer profile that failed to fulfilled the guarantees. However, current PPI modelling approaches do not support any metric that involve information related with the human resources that performed the task.

**Metrics that involve different processes** Some SLA metrics have to be defined over two or more process instances. This happens when a metric require execution information from two different processes to be computed. An example was found in one of the analysed BPO SLAs, where the number of incidents post production had to be obtained, and this metric required information from the incidents and the software delivery processes. Again, this metric cannot be modelled using current PPI modelling approaches, since a PPI focuses on just one process by definition.

These limitations could be easily addressed in PPINOT just by doing minor changes in its metamodel. Complex time metrics are supported just by defining filters over *time measures*, so that idle time, suspend time, calendars or timetables can be taken into account when computing the time for the measure; and by adding a new type of measure, *time instant measure*, that measures the date and time in which an event takes place instead of the duration between two events.

However, others are left as future work since they require more significant changes. The metrics that involve human resources can be partially addressed using an extension to PPINOT to define resource–aware PPIs [43]. Finally, the metrics that involve different processes can be defined as a *derived measure* that relates measures in each process instance, but it is necessary to include information on how to correlate process instances when defining them, which is something that will be addressed in future work.

### 4.5.3 RQ3: Difficulties modelling SLAs

After modelling the iAgree documents and the 52 metrics with PPINOT, we found some limitations concerning the definition of metrics, whereas iAgree and the models used to define business processes, SLOs, penalties, and rewards proved to be capable to model all possible situations.

The application of the proposed approach for defining SLAs of BP services to real scenarios showed up that the parameterising of the guarantee has to be carefully analysed. The scope has provided to be very useful in several SLAs, as SLOs are defined once but they are applied to multiple scopes. Specifically, when the domain of a parameter is not completely known before defining the SLOs it would not be possible to define a single SLO for each parameter value (e.g.: Node variable in the example).

## 4.6 Related Work

Regarding the analysed proposals for the modelling of SLAs in chapter §3, we compare the proposals relevant for the modelling of SLAs for BPs. In Table §4.6, we compare the significant features in our scenario. That is, the set of attributes required to model it. First, we identify if they provide a vocabulary for the modelling of non-computational services or just computational, including the description and also the performance metrics. Then, we check if they provide a template model, as WS-

Agreement, to derivate new SLAs from a template. We also evaluate if they define compensation functions to penalize or reward overfulfilling or overfulfilling of SLOs. And last, we check if they provide tooling to support the design and creation of such SLAs.

Among the analysed proposals, we have generic purpose proposals, together some of the domain specific ones. As specific domain proposals, we find Daly [34] *et al.* , which propose a simple SLA language for CRMs. They describe metrics as customer satisfaction or utility, as the relationship between accomplishment and customer satisfaction so they are ad-hoc defined for the proposed domain. Augenstin *et al.* [7] propose a language to describe logistic services. They mainly focus on functional aspects. The performance quality is named, but no any attribute or related language is proposed. They propose to model service descriptions from a BPMN using a model to model transformation. Bar-Isaac *et al.* in [10] 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). The domain is specific to the scenario proposed and it cannot be extended. Cardoso *et al.* [25] propose a description language for services that includes business characteristics together with technical or operational parameters. Unlike our proposal of managing a business process as a service to define their SLA, this work is focused on managing services including business perspective.

There are a number of proposals which are independent of the domain, but they do not address business processes or non-computational domains. These are SLA* [93], which is a domain independent language generalizing other specifications such as WS-Agreement, WSLA or WSDL. The document structure is similar to WS-Agreement and it can be extended with the requirements of each domain. However, as far as we know, there is not domain proposal to describe non-computational services. Rule-Based Service Level Agreement Language (RBSLA) [143] supports specific vocabularies defined externally. They provide an SLA document structure supporting guarantees with compensations but there is no proposal with a business domain vocabulary to validate the proposal. Forster *et al.* [63] propose a model for business processes and quality requirements as temporal logics problems. However, they do not handle with other kind of measures and they do not provide a compensation language. Sauvé *et al.* [161] propose a methodology to calculate SLO thresholds to sign IT services SLAs according to service function cost from a business perspective. However, this proposal is focused on the cloud domain and how the cloud performance metrics relates to business costs. Kieninger *et al.* [95] describe a categorization of IT services and outline a mechanism to obtain efficient SLOs for them. However, they do that in a conceptual level and do not detail how they can be formalised to enable their automated management. Therefore, the service description and SLOs are limited to the proposed services.

And among the proposals which specifically address business processes description and metrics are Linked-USDL Agreement [70], which specifically addresses non-computational and computational domains. They use ontologies to describe service

| **Proposal** | **BP** | **Metrics** | **GT** | **Template** | **Comp.** | **Tooling** |
|---|---|---|---|---|---|---|
| SLA* [93] [34] | x | x | ✓ | ✓ | ✓ | ✓ |
| Linked-USDL Agreement [70] | ✓ | ✓ | ✓ | ✓ | ✓ | x |
| CPPA [135] | ✓ | ✓ | ✓ | x | x | ✓ |
| RBSLA [143] | x | x | x | ✓ | x | ✓ |
| Daly [34] | x | x | ✓ | x | x | ✓ |
| Augenstin [7] | ✓ | x | ✓ | x | x | ✓ |
| Nguyen [133] | ✓ | x | x | x | x | ✓ |
| Chau [29] | ✓ | ✓ | ✓ | x | x | ✓ |
| Forster et al. [63] | x | x | ✓ | x | x | x |
| Sauvé [161] | x | x | ✓ | x | x | x |
| Kieninger [95] | x | x | ✓ | x | x | x |
| Cardoso [25] | ✓ | x | x | x | x | ✓ |
| Bar-Isaac [10] | x | x | x | x | ✓ | x |
| Our Proposal | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 4.4: Metric definition models for BPs

and metrics together with constraints for the guarantees. However, the scenarios modelled are syntetic and they do not provide tooling support to monitor SLAs. Nguyen *et al.* [133] propose aligning the SLAs for the different services/tasks with different layers: business, people and technology. So they describe different categories for SLOs and for SLA in the different layers and check how they align with the other layers. However, they do not provide a generic language for service description in non computational domains. Collaboration-Protocol Profile and Agreement Specification (CPPA) [135] (which was published in 2002), which complements ebXMLBP for commerce collaboration. They focus on electronic collaboration so they provide a specific language for electronic tasks, such as document exchange protocol, message system, secure communications, etc. Chau *et al.* [29] propose to define performance metrics and goals based on business processes. they relates SLAs and business process artifacts where guarantees over the process are defined through process events.

The namely attributes in the table are: (i) *BP*, if the proposal enables to describe the non-computational services and business processes or it is related to a specific computer domain, (ii) *Metrics* if they provide performance generic metrics for non computational services (if they are defined ad-hoc, we considered that they do not support non computational metrics), (iii) *GT* if they provide a language to describe guarantees over metrics in non computational services (iv) *Template*, if they consider the creation of SLAs derived from a Template, (v) *Comp.*, if they provide compensations definitions and (vi) *Tooling*, if they provide tooling support to create SLAs.

As we can see in the comparison table, just some of the proposals provide a language to describe non-computational services or business processes and, amongst them, only three [70],[135],[29] provide a language to describe metrics over them. Furthermore, almost none of the agreement models consider the definition of compensations.

Therefore, in summary, none of the proposals cover the required expressivity for our model, including service description, performance metrics and compensations, neither give support to the lifecycle.

## 4.7 Summary

With this proposal, BP SLAs can be modelled by combining mechanisms for modelling computational SLAs with mechanisms to model business processes and PPIs.

Furthermore, our proposal to model SLAs combines well founded approaches and standards for modelling computational SLAs and PPIs. Specifically, we rely on WS-Agreement [6], which provides the general SLA structure, BPMN [138], which is used to model the business process related to the service, PPINOT [41], which allows the definition of metrics, and iAgree [126], which provides a language to define SLOs and penalties.

The application of the proposed approach to a number of real scenarios allowed us to conclude that our approach is able to model all possible situations in these scenarios except for some identified limitations concerning the definition of SLA metrics. Some of them could be solved by applying minor changes to the PPINOT metamodel. However, other limitations require more significant changes that shall be carried out in future work.

# Management of SLA-Driven Business Processes

*Never promise more than you can perform.*

*Publilius Syrus ( 85 BC - 43 BC),*

## 5.1 Introduction

In the previous chapter, we discussed the model to describe SLAs for Business Processes. There are a number of tools to support the execution of business process (or are aware of process flows). However, although some current existing PAISs (Process Aware Information Systems) support multiple customers (*e.g.* BonitaSoft[1]), they do not provide facilities to manage their SLAs as a first level citizen. In this chapter, we analyse how to provide automated support to the SLA lifecycle in PAISs. A PAIS with SLA–aware capabilities, i.e. a PAIS that manages explicit definitions of SLAs, can enable the sharing of monitoring mechanisms, process configuration [179] or optimise the management of shared resources used by different parties.

We focus on the late phases of the SLA lifecycle (described in Subsection §3.3.2) as the SLA creation phases have been deeply analysed by Resinas [153], Pablo Fernandez [60] and Muller [122]. Regarding these stages, after reviewing the scenarios, we analyse the impact of *deployment* of the SLA in order to describe how (i) **configure** the service according to the SLA and then we identify the requirements to (ii) **monitor** the accomplishment of SLA during the *fulfillment* phase. Then, we propose how to extend PAISs to support these operations through SLAs.

On the one side, regarding **monitoring**, we propose extend an SLA–aware system to instrument the monitoring of the metrics for different parties and SLAs. Doing so, we can provide different dashboards for each customer, aggregated stats for the system provider, or create alerts based on the accomplishment of the SLOs. On the other side, regarding **configuration**, we describe an architecture to extend a BPMS that configure the process execution to meet the SLA terms. We specifically consider BPMS platforms because the process execution depends on the supporting system. This architecture

[1]`http://www.bonitasoft.com`

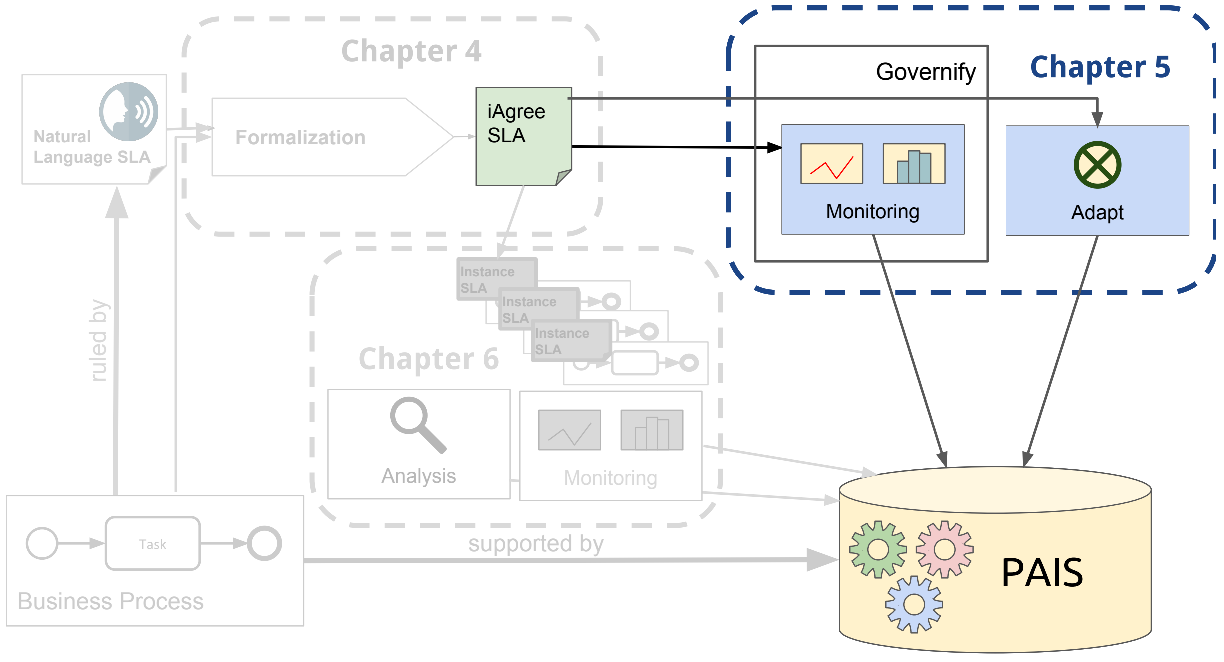


Figure 5.1: Contributions background

has been developed as a prototype in Open Source BPMS Camunda. Other operations could be considered during the deployment and fulfillment phases, such as optimization, but it is out of the scope of the current dissertation, although some initial results are included in Appendix §B.

The contribution of this chapter regarding the dissertation is highlighted in Figure §5.1.

In the previous chapter, we focused on the formalisation of SLAs such as the one described in the STR document proposed by APC for IT maintenance outsourcing, that is then discussed with the potential contractors. As a result, the required performance goals (e.g. maximum times to solve different tasks) or the service execution options (e.g. executing documenting task or human assignments for certain tasks) are negotiated with each contractor and different agreement offers are proposed which result in different final agreements with each contractor (after the public contract tendering). The agreement offers have to be consistent with the STR, which means they should guarantee similar or even more restrictive performance terms but they can differ from one contractor to another. The required goals and related penalties are again depicted in Figures §5.1 and §5.2.

In this scenario, the APC provides different tools which act as PAIS to support the task execution for all the contractors and capture all the service relevant events (plan interventions, start of related tasks, ...). As the terms agreed with each contractor vary, when the SLAs are deployed, the supporting tools have to be aware to their different

| **Criticality** | Critical | High | Mild | Low |
|---|---|---|---|---|
| **Response Time** | 0.5h | 2h | 5h | 5h |
| **Presence Time** | 4h | 8h | 30h | 60h |
| **Solving Time** | 2h | 4h | 6h | 8h |
| **Doc. Time** | 4h | 12h | 24h | 48h |
| **Timetable** | Office | Office | Office | Office |
| **Calendar** | Local | Local | Local | Local |
| **Document required** | N/S | N/S | N/S | N/S |
| **PlanFI HR** | N/S | N/S | N/S | N/S |

Table 5.1: Committed terms for the FI Service SLA

| **AFIP** | **Penalty** |
|---|---|
| $94\% \leq$ AFIP $< 95\%$ | -1% |
| $93\% \leq$ AFIP $< 94\%$ | -2% |
| $92\% \leq$ AFIP $< 93\%$ | -3% |
| $91\% \leq$ AFIP $< 92\%$ | -4% |
| $90\% \leq$ AFIP $< 91\%$ | -5% |
| AFIP $< 90\%$ | -10% |

Table 5.2: Penalties definition (in monthly billing percentage) for the FI Service SLA

terms so they are instantiated and executed properly for each party (e.g.: executing the documenting task only when it is required for a contractor).

Once the SLAs are deployed and service instances are executing, the APC evaluates their fulfillment. This evaluation of the SLAs requires retrieving performance information and processing it regarding the involved party and their targets (i.e.: monitoring guarantees, providing resources to achieve goals, etc.). And, lastly, the common management of the different processes could be used to optimise the process execution.

## 5.2 SLA Monitoring

The monitoring information is useful for different purposes. First, to evaluate the accomplishment of the SLOs and apply penalties in case they are not fulfilled. Second, to provide visual dashboards for managers and business domain experts. Then, a system of alerts based on the monitoring indicators could be provided. And last, they would be required to make decisions about optimizing service execution (e.g. resources management) in runtime. Regarding this last purpose, the monitoring information is the basis for systems that predict the evolution of performance supporting the decision making.

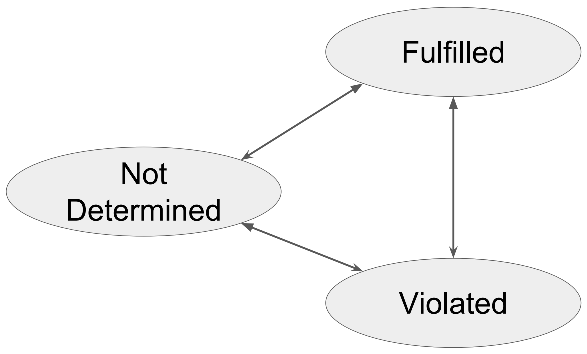


Figure 5.2: Guarantee Term states in WS-Agreement

In order to describe this operation we, first, formalize its definition:

**Definition 5.1 - Monitoring Operation.**
*Let SLA be the SLA to monitor, M the performance information from the supporting systems, T a time period and ST a set of states of the SLA. Thus, we define Monitoring as a function which for each value of SLA, M and T returns a set of states ST of SLA for that time period T.*

The relationship between performance information, *M*, and the guarantees is described with the metrics. The expressiveness of a metric is related to, on the one side, the metric domain, that is, if we can use just built-in metrics in the PAIS (e.g: Metric #SolvingTime: Duration of a single Task) or they can be defined in a generic way for business process domain (e.g.: Metric #SolvingTime: Duration from the status *Start of Task #1* to the status *End of Task #2*). On the other side, related to math, logic or aggregated operations, that is, if we can define metrics deriving a base metric with additional operators, such as a Metric that calculates the Average of the Metric #SolvingTime for all the process instances. Existing solutions do not provide generic definitions for the business process domain or more complex operators.

### 5.2.1 Status of the SLA

The aforementioned status is referred to the SLA status introduced in the Section §3.3.2. The status of the SLA is monitored during the Observed phase and it depends on the state of its guarantees, which possible states are again depicted in Figure §5.2.

The monitoring operation retrieves the performance information to compute the metrics included in the guarantees. Once the metric value is obtained, then it has to be evaluated regarding the SLO. If the value accomplishes the target included in the SLO, then the SLO and its corresponding guarantee are considered in the status *Fulfilled*. *Fulfilled* is one of the possible Guarantee Term states depicted in Figure §5.2. If the SLO is not accomplished, then the guarantee is considered *Violated*. When the guarantee is violated, the compensations defined for that guarantee have to be applied. In the Figure §5.3 we describe different examples of SLA status depending on the performance value.

In practice, the scope in the guarantee multiplies the number of guarantee evaluations per the scope domain. In the example in Figure §5.3 (A), the scope *office* has 4 possible values: Huelva, Sevilla, Cádiz, Córdoba so the metric has 4 different values, one per scope value, and the guarantee behaves as 4 different guarantees, with their own status.

Regarding the guarantee window, it also impacts how the status of the guarantee is evaluated. In the example in Figure §5.3 (B), with a static monthly window, the metric value, 96.9%, for January does not accomplish the SLO objective so the Guarantee is *Violated*, but in February, the status is reset (*Not Determined*) until the month finishes. Then, when the month finishes, it is evaluated again and then the metric value is 99.9%, so the SLO is accomplished and the guarantee is *Fulfilled*. Therefore, a guarantee with a scope with 10 possible values and a static monthly window, will require 120 evaluations of its state in a full year (12 months X 10 scope values). For the sake of simplicity, we do not consider in this discussion or in the Figure §5.3 that the guarantee can be evaluated *Violated* or *Fulfilled* even before the defined Window finishes (e.g.: in an SLO of 99.9% of Availability in a month and the service has been unavailable more than 0.1% of the month duration, no matter what happens during the rest of the month, the SLO will not be accomplished) but it has to be properly evaluated in any moment.

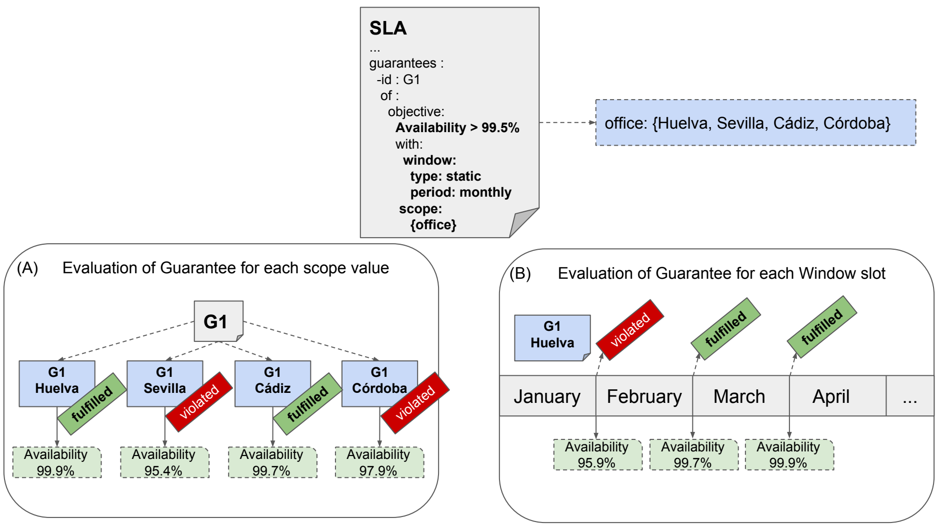


Figure 5.3: Metric values and Status of Guarantee Terms

In the Appendices, Subsection §C.2.1, we list the metric computers for this agreement.

## 5.2.2 Infrastructure specification

Business processes are usually supported by different tools to handle user interaction, orchestrate tasks, automate processing, etc. Although there are BPMS to manage generic BP, each system has their own information management and storage. Some tools, such as the Camunda BPMS, provide an API to query executed events and processes. In other cases, the information is accessible reading the executed event logs or even only through querying database. Specifically, in the example scenario different tools are used to support process execution. So, in order to properly evaluate the performance, it is required to include information to handle this tooling heterogeneity. Our proposal is using an homogenous event log that is fed by each supporting tool and process this log. In the following excerpt, we depict the sections in the context of the SLA that include the URL of the event log in the Infrastructure section and some parameters about the log structure. Amongst these parameters are the event attribute corresponding to the id of the process instance, *instanceIdColumn*, the name for the process event (which task, gateway, etc.), *actionColumn*, or how to identify the event that corresponds to the end of the instance, *terminator*.

```
context:
  ...
  infrastructure:
    ...
    logs: 'http://naos.logs.chap.governify.io/api/v1'
  logs:
      naos:
        default: true
        uri: 'http://naos.logs.chap.governify.io/api/v1/logs'
        stateUri: 'http://naos.logs.chap.governify.io/api/v1/count'
        measures: ''
        structure:
          instanceIdColumn: INCIDENT_ID
          timestampColumn: CREATION_DATE
          actionColumn: SID
        terminator: >-
          [{"column": "ACTION","values": ["SDK_HIST_TRANS_PHASE"]},{"column":
          "VALUE","values": ["Cierre definitivo"]}]
```

We have to remark here that when the log is processed only the finished instances are considered, because, in some cases, the intermediate events can produce errors in the measure because as long as the instance is running, the supporting tools enable to go back in the process to previous stages so, if we consider these instances, values previously processed could change.

### 5.2.3 Monitoring component

We propose the architecture depicted in Figure §5.4 for the processing of performance. We assume that there is a REST API (Performance Information in the Figure §5.4) where to connect for a batch query or to receive an event stream. After the processing, the monitored data are also provided as a service (SLA Monitor API in the Figure §5.4). The components and their responsibilities in this architecture are:

**SLA Monitor API**. The API just exposes the operations related to the monitoring of an SLA, that is, given an SLA, returning the result of evaluating their fulfillment of its SLOs and the timeline of metric values. This evaluation is performed by the **SLA Monitor** component.

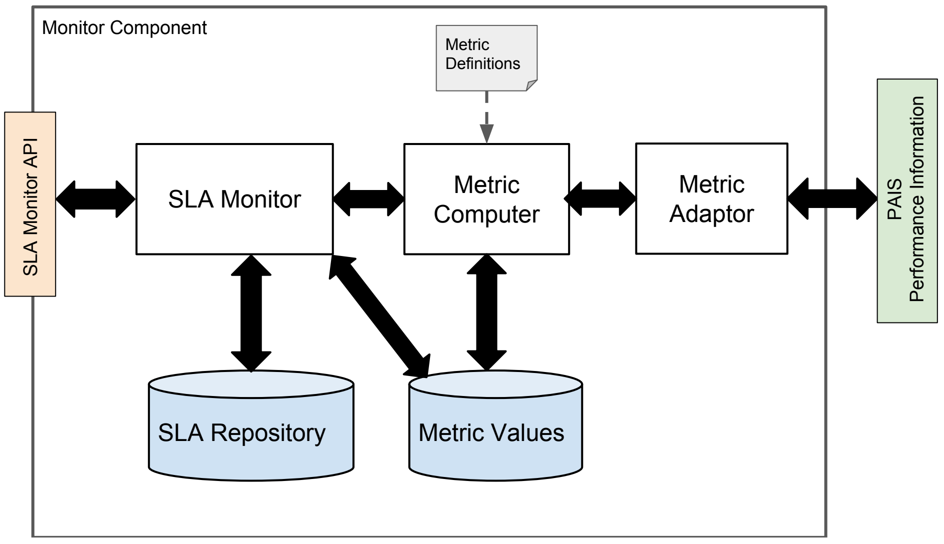


Figure 5.4: Monitoring Architecture

**SLA Monitor**. This component processes SLA terms, which include SLOs with metrics and target values. The metric values are retrieved from the **Metric Computer**. and this component uses them to evaluate their SLOs accomplishment regarding their targets to determine the status of Guarantee Term in the SLA. This component also applies the possible compensations if the Guarantee Term is violated.

**Metric Computer**. This component encapsulates the processing and storage of metric values with both batch and runtime processing (it can be configured). In the stream processing model, the temporal monitoring information is stored in a repository of **Metric Values**. The metrics to process are described with the Metric Definitions related to an SLA. When the SLA Monitor requests the evaluation of a metric, this component receive performance information from the PAIS with the **Metric Adaptor** and process it according to its *Metric Definition*.

**Metric Adaptor**. This component makes the monitoring independent of the mechanism provided by the PAISs to retreive information. The adaptor can either connect as listener to receive events as they are dispatched from the PAIS either make a query to the API provided by the PAIS to query a set of events.

In the following subsections, we exemplify the two implemented approaches for the type of processing. The method to choice depends on supporting infrastructure capabilities or performance criteria, as both approaches are similar in terms of expressiveness.

### 5.2.4 Alternatives for Monitoring

In this scenario, we focus on the definition and development of PAIS independent operations to compute measures over business process executions. To achieve this, we use the set of tools and techniques defined for PPINOT [40]. Computing measures as non-intrusive methods depends on supporting infrastructure and the facilities to retrieve information from it.

Regarding the processing of performance data, there are usually two different approaches to process them: (i) Batch and (ii) Stream processing. Both approaches are compared in Figure §5.5 for the processing of average time of process instances. The performance information is depicted with events of start (green colour) and end (red colour) of different process instances (PI #1, #2, ...).. On the one side, the batch processing is performed capturing at once all the performance. In the upper side of Figure §5.5, there is batch capture of five events, the start of process instances #1, #2 and #3 and the end event of process instances #1 and #2. As the instance #3 is not finished, is not used for the average processing (although it could be used considering the current time). We obtain the average with the normal processing: the different between the sum of end event times and the sum of start event times, and dividing the result by the number of instances (e.g. two). On the other side, the stream processing is performed as the events appear sequentially. In the lower side of Figure §5.5, the stream processing is depicted. When the first event in the stream appear, i.e. the start event of process instance #1, it is processed. As there is no information about the end time, no duration can be calculated and it has to be internally stored until the related end event appears. Then a second start event from process instance #2 appears that has to be also internally stored. Then, the end event of process instance #1 comes and the duration of process instance can be processed as the first average time. And this processing continues as the events appear. Batch processing can be demanding on CPU, depending on the amount of data processing but it can be requested on demand, as for example, daily at 0.00 when the work load is low. On the opposite side, stream processing usually requires less computation as the events are processed iteratively one by one, but it requires a constant use of resources, such as memory to maintain the processing state. These approaches are detailed in the next subsection §5.2.3, describing their implementation.

We can also consider the mechanism provided to extract performance information from the PAIS, that can be based on querying an API or event listening. The mechanism could be related to the processing approach but they are independent. We can perform a query to the API to get a set of execution events but store them in a log instead of processing as a batch, and later processing the log as a stream. And, reversely, we can listen events and process them directly as a stream, or store them and process them later as a batch.

#### Batch processing in a BPMS: Camunda

BPMSs usually provide an interface to query performance information. This is the case of Camunda API, which offers methods to retrieve any activity (task, gateway or event) information so we can easily get timestamps for time indicators or account tasks executions with the Camunda History Service.

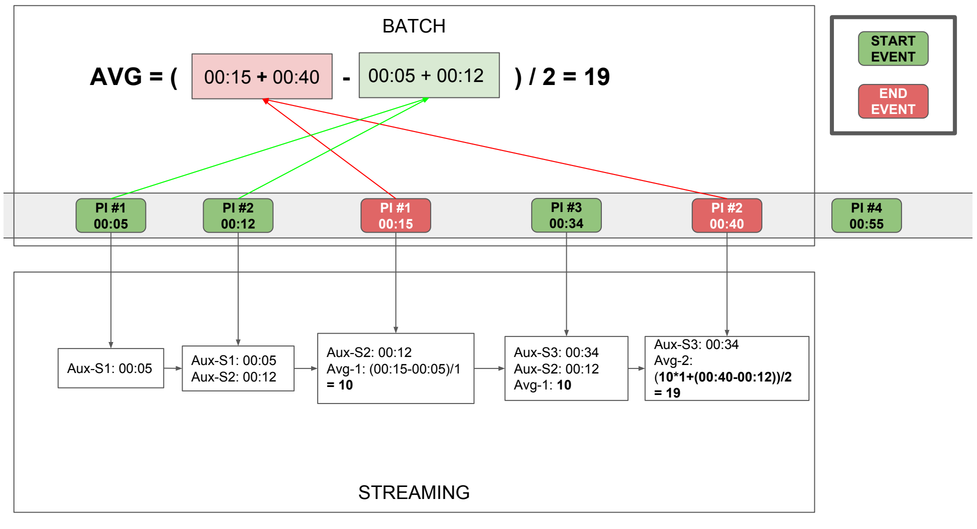

BATCH
AVG = ( 00:15 + 00:40 - 00:05 + 00:12 ) / 2 = 19
START EVENT
END EVENT
PI #1 00:05
PI #2 00:12
PI #1 00:15
PI #3 00:34
PI #2 00:40
PI #4 00:55
Aux-S1: 00:05
Aux-S1: 00:05 Aux-S2: 00:12
Aux-S2: 00:12 Avg-1: (00:15-00:05)/1 = 10
Aux-S3: 00:34 Aux-S2: 00:12 Avg-1: 10
Aux-S3: 00:34 Avg-2: (10*1+(00:40-00:12))/2 = 19
STREAMING

Figure 5.5: Stream vs Batch Processing

The Batch example illustrated in Figure §5.5 to compute the average Time Measure for process instance duration, is implemented requesting to the Camunda History API, the events *start* and *end*, from the proper process instances. As we get all the process instances that have finished, we just sum all the timestamps for event *start* on the one side, the timestamps for event *end* on the other side, then substract the former to the latter and, lastly, divide the result by the number of instances.

### Stream processing

In other scenarios, the stream of events processing is used. So we process the events as they are received, filtering the activities for metrics of interest. The processing of average Time Measure for process instance duration is calculated as we listen the events *start* and *end* and follow the processing depicted in Figure §5.5. When we receive an event *start* for a process instance #i, it has to be registered without processing until we receive the event *end* of that process instance #i. Then we calculate the instance duration (*end* event time - *start* event time), and process the average, considering the previous average and number of process instances computed.

In real scenarios with massive process instances, the processing of metrics in both approaches can degrade performance, either because memory requirements either because performing a massive number of metric processing computing at once. Nowadays, improving big data processing is a research goal so a number of techniques and tools have been developed focusing on this topic. We take advantage of these techniques to enrich our proposal [87].

On this regard, we use incremental calculation technique. This technique is implemented by a number of tools to provide scalable processing in big data scenarios (such as Apache DataFu over Apache Hadoop). For the introduced metrics, we have to calculate average times on static 30 days window. A full month measure requires processing full data for 30 days. However, if we want to monitor this measure daily, we can store intermediate times so future processing can be made faster if we consider only the different day intervals. Therefore, if we monitor one week and calculate average for that week, monitor 3 days later only requires adding data for these 3 days. Incremental calculation is described below.

Let InstDay$_i$ the number of process instances in a day $i$ and TimeDay$_i$ the accumulated Develop Time for all the process instance in a day $i$:

$$InstDay_i = N \tag{5.1}$$

Total Develop Time per Day:

$$TimeDay_i = \sum_{j=0}^{N} DevelopTime_j \tag{5.2}$$

Accumulated Process Instances in the last 7 days:

$$AccumInstDay_i = \sum_{j=i-7}^{j} InstDay_i \tag{5.3}$$

Accumulated Develop Time in the last 7 days:

$$AccumTimeDay_i = \sum_{j=i-7}^{i} TimeDay_j \tag{5.4}$$

Average Develop Time in the last 7 days:

$$AverageTimeDay_i = \frac{AccumTimeDay_i}{AccumInstDay_i} \tag{5.5}$$

If we store intermediate accumulated and daily number of instances, and total process time for the given day $i$, we can incrementally calculate the average for the following day $i$+$1$:

$$AccumInstDay_{i+1} = AccumInstDay_i + InstDay_{i+1} \tag{5.6}$$

$$AccumTimeDay_{i+1} = AccumTimeDay_i + TimeDay_{i+1} \tag{5.7}$$

$$AverageTimeDay_{i+1} = \frac{AccumTimeDay_{i+1}}{AccumInstDay_{i+1}} \tag{5.8}$$

This process is depicted in Figure §5.6, where a window time is computed for the first 3 days and intermediate results are stored (Intermediate Stage 1). In our example, these intermediate data are accumulated process time and accumulated number of instances. On the fourth day, instead of computing full measures for previous days, we reuse previous results and add the day 4 execution data and get the new result. So, measure calculation leverages this technique to simplify queries, and avoids unnecessary data processing.

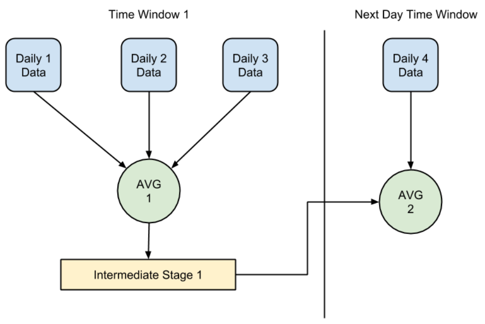


Figure 5.6: Time Window calculation

### 5.2.5 Applicability of our proposal

This proposal has been applied on two real projects of transfer of technology to provide the computation of more than 20 KPIs described by iAgree and PPINOT independently of the supporting tool for the processes. These projects were in collaboration with the Andalusian Health System, AHS, and Andalusian Finance Council, AFC.

A monitoring component has been developed following the event stream approach based on querying a log. This component has been deployed in the Governify [2] platform, which already provides different visualizations for REST API monitoring. The Metric component was developed in Java while the rest of the platform is in Node.js. The dashboard acts as the consumer of the endpoints defined for the metric schemas (defined in Previous Chapter §4). By doing so, Governify provides a complete ecosystem to: (i) create SLAs with iAgree syntax with some analysis operations to avoid manual errors and (ii) monitor the processes performance with the proper metric schema and infrastructure information included in the SLA. The code of the platform is published in Github: `https://github.com/isa-group/ppinot`.

The supporting PAISs are heterogeneous tools with heterogeneous data storage and mechanisms to retrieve performance information, such as Trello, JIRA or Dropbox.

[2] `http://www.governify.io`

Therefore we developed wrappers for the different tools to export the corresponding performance information to a log storage. Therefore, in this scenario, the metrics defined in the SLAs are shared with these wrappers to export only the relevant information. This architecture is displayed on the Figure §5.7, that depicts the integration of wrapped tools together the SLA processor for the AHS dashboard.

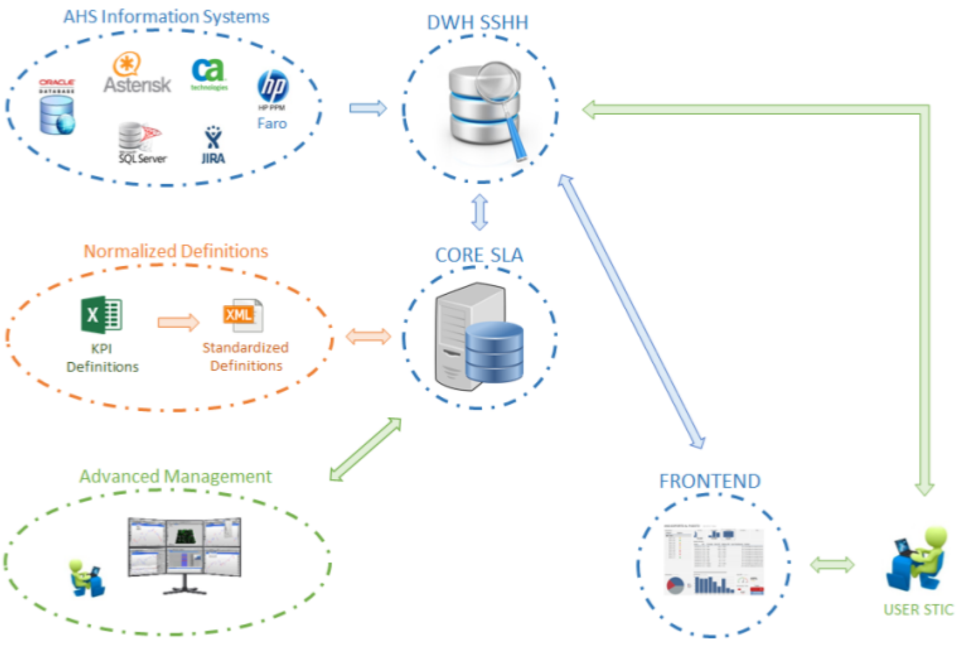


Figure 5.7: Log processing

In our component, the Metric Adaptor processes this log as a stream which is processed by the Metric Computer with the metric definitions from the SLA. In the prototype, the log included millions of events (with Gigabytes of size), which were filtered by monthly periods because the metric definitions. The SLA included 12 KPIs, so considering the different scopes for each KPIs (the scope domain multiply the number of computations) so in a normal month we processed half millions events for a total of 3000 KPI computations. A graphic dashboard is displayed with the evaluation of the SLA in the Figure §5.8, where we blur the text because confidentiality issues. This graphic component requires some additional information to configure the graphic widgets. An excerpt of this information is described in the Table §5.3. The fields required are: (i) *Title*, the displayed Name, (ii) *Name*, the identifier of the data, (iii) *Mandatory*, if the value is always displayed, (iv) *Format*, to indicate the data type and (v) *Description* to provide a human readable description of the data.

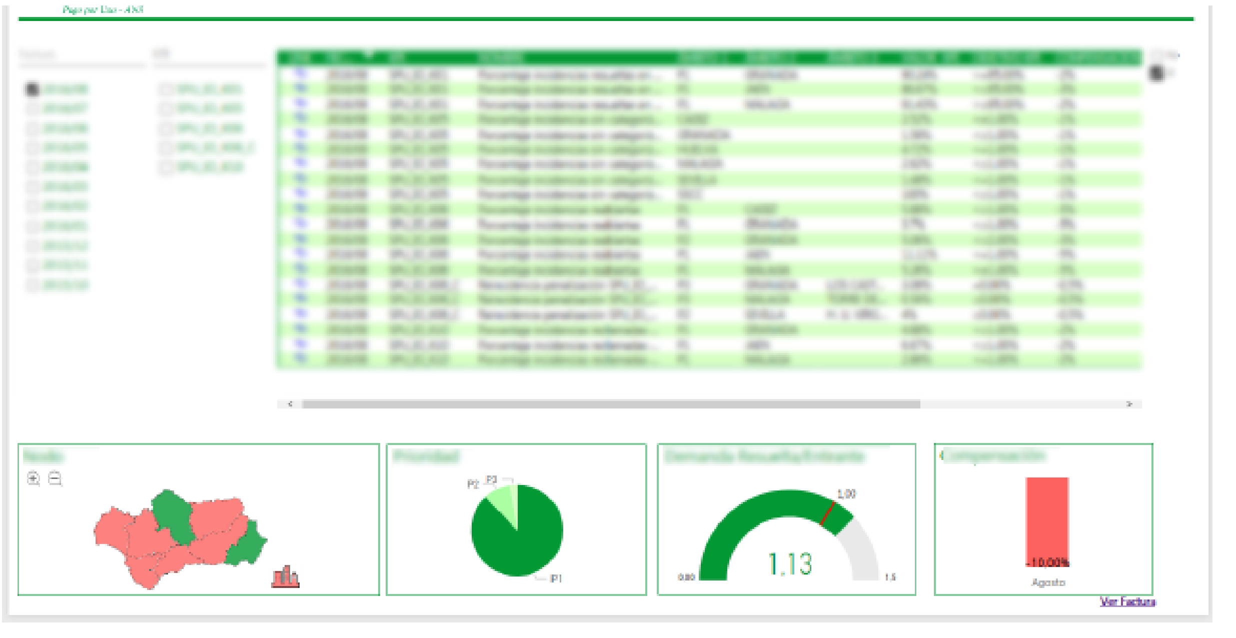


Figure 5.8: Project Dashboard

| Title | Name | Mandatory | Format | Description |
|---|---|---|---|---|
| Unit Value KPI | UNITKPI | YES | Text | Unit used to measure KPI |
| Frequency | FREQUENCY | YES | Integer | Frequency to measure months |
| Init cycle Day | INITDAY | YES | Integer | Day to start the cycle |

Table 5.3: Excerpt of Widgets configuration

### 5.2.6 Related Work

There are a number of systems that can support the management of business processes, including monitoring. However, these systems usually manage business processes instances from different parties as independent models and instances and provide a simple support for SLAs such as defining sets of simple event metrics directly on a dashboard.

In the service of our scenario, the supporting systems are heterogeneous for each party and do not provide SLA-Aware monitoring mechanisms. Even in existing BPMSs the monitoring mechanisms to store or log common process events, such as Task Status changes (with Business Activity Monitor, BAM, or similar) do not process more complex indicators, such as our example SolvingTime and have to be externally defined and processed. Furthermore, existing BPMSs commonly monitor performance with ad-hoc metrics while our proposal with PPINOT is generic, which is specially useful when we have to use an heterogeneous number of tools as in our scenario.

Regarding non-computational services, some of the models introduced in previous chapter, [93], [70], [29], [63] are considered a first step to automate the management of non-computational SLAs but they do not propose any monitoring architecture or mechanism.

There are also a number of proposals related to the monitoring of services which are usually related to Cloud platforms. Amongst these, we find X. Li *et al.* [108], who define a mechanism to monitor Trust over the Performance of cloud computing services, guided by SLAs. This proposal is specific for cloud computing and they do not address how the performance is retrieved. rSLA [171] is a language for specifying, monitoring and enforcing SLAs for cloud services. They discuss about the advantages of using conditions and actions for the definition of guarantees and compensations but they do not address how the monitoring is performed. Emeakaroha *et al.* [58] propose to map performance abstract measures that appear in SLA, like Availability, to specific monitoring variables, like Server Downtime to measure them without ambiguity but they do not specify how the real measures are obtained. Brandic *et al.* [19] also propose a mapping from abstract SLA metrics to the real taken measures from cloud infrastructure but they do not specify how the real measures are obtained.

In order to monitor domain independently, Leitner *et al.* [104] propose monitoring SLAs in runtime to predict SLA violation in simple and composed services. Their engine use Complex Event Processing of event streams to perform the evaluation of performance metrics. They support complex measures but they do not provide a domain language for business processes.

And as an example of BPMS, Bonita BPM suite [3] provides its own performance monitoring system based on activities events (Business activity monitoring, BAM), such as event duration but without generic complex indicators (e.g.: sum of (a) average Time Measure from Start of a Task #2 to End of a Task #1 and (b) average Time Measure from Start of a Task #4 to End of a Task #3).

In the table §5.4, we analyse the comparison of these proposals related to monitor the performance and SLA. To compare the proposals, we consider first the *Domain* where they apply, computational (**x**) or non computational (✓) domains. Then we evaluate their *Expressiveness*, i.e., if the proposal considers just event metrics (**x**), such as a single Task duration, or they provide more complex metrics, as operating with a set of event metrics or involving different events for a metric, such as the duration from the start of an event to the end of another event (✓). Regarding the processing *Mechanism*, we check if they support *batch* processing, *stream* processing, *both* or *unknown* if the monitoring proposal only discusses the evaluation of the metric but not the developed approach. As different SLAs can share the metrics, we evaluate if the proposals support monitoring *Multiple* SLAs from the same metric definition or not. And, lastly, we evaluate if the proposal applies or not some *Compensation* regarding to the result of evaluating the monitored SLOs.

As a result of this comparison, we can highlight that while some of the proposals provide the definition of complex metrics [58] or [19], they do not address how extend

[3] `http://www.bonitasoft.com/`

| Proposal | Domain | Expressiveness | Mechanism | Multiple | Compensation |
|---|---|---|---|---|---|
| X. Li [108] | x | x | unknown | ✓ | x |
| Bonita BPM suite | ✓ | x | both | x | x |
| Leitner [104] | ✓ | ✓ | stream | x | x |
| rSLA [171] | x | x | unknown | x | ✓ |
| Emeakaroha [58] | x | ✓ | unknow | x | x |
| Brandic [19] | x | ✓ | unknown | x | x |
| Our Proposal | ✓ | ✓ | both | ✓ | ✓ |

Table 5.4: Expressiveness of SLA Monitoring

a system to retrieve them. And, reversely, while some proposals handle the processing of monitoring performance (Bonita BPM or [104]), they not provide a generic language to define complex metrics. Both perspectives are addressed by our proposal.

## 5.3 BPs configured by SLA

When different contractors sign different SLAs with the APC, the SLA terms can vary not only in guarantees but also in the service execution. The **model** to describe the service business process has to support the process configuration to provide this variability likewise the supporting PAIS has to be aware of this variability to **configure** the deployed or executed business processes for each party. As WS-Agreement templates aim to provide a common document for the SLAs with different parties, we can use it as the document to model the STR. Tables §5.5 and §5.6 describe the committed service terms by two contractors, A and B, to the APC. SLA with *contractorA* commits more restrictive times to solve incidences than the STR and requires documenting Critical and High Severity interventions, while SLA with *contractorB* requires even more restrictive times to perform tasks, documenting all the issues and requires that task **Plan FI** can only be performed by person with the manager role. These differences imply, on the one hand, that the PAIS has to support these multiple parties, e.g.: to perform or not the documenting task depending on the contractor SLA, and, on the other hand, monitor consistently the performance of their related process instances to detect SLA violations, take preemptive decisions, etc, which was addressed in the previous section.

In this section, we address the adaptation of process execution to the different parties. In order to describe this operation we, first, formalize its definition:

**Definition 5.2 - Configuration Operation.**
*Let $SLA_t$ be an SLA template for BPs and C a configuration for such BPs. Thus, we define Configuration as a function which for a value of $SLA_t$ and C returns a configured version of the BPs in $SLA_t$ for that configuration C.*

| **Criticality** | Critical | High | Mild | Low |
|---|---|---|---|---|
| **Response Time** | 0.5h | 2h | 5h | 5h |
| **Presence Time** | 4h | 8h | **15h** | **30h** |
| **Solving Time** | 2h | 4h | 6h | 8h |
| **Doc. Time** | 4h | 12h | 24h | 48h |
| **Timetable** | Office | Office | Office | Office |
| **Calendar** | Local | Local | Local | Local |
| **Document required** | **All** | **All** | N/S | N/S |
| **PlanFI HR** | N/S | N/S | N/S | N/S |

Table 5.5: Committed terms by the Contractor A for FI Service SLA

| **Criticality** | Critical | High | Mild | Low |
|---|---|---|---|---|
| **Response Time** | 0.5h | 2h | **4h** | **4h** |
| **Presence Time** | 4h | 8h | **15h** | **30h** |
| **Solving Time** | 2h | 4h | 6h | 8h |
| **Doc. Time** | 4h | 12h | 24h | **24h** |
| **Timetable** | Office | Office | Office | Office |
| **Calendar** | Local | Local | Local | Local |
| **Document required** | **All** | **All** | **All** | **All** |
| **PlanFI HR** | Manager | Manager | Manager | Manager |

Table 5.6: Committed terms by the Contractor B for FI Service SLA

This configuration depends on, on the one side, different agreed service terms, such as the Documenting Task in the FI service, which is always executed for Contractor B (Table §5.6), while depends on the intervention severity for Contractor A. And, on the other side, the process provider can manage different process models or instances for two different customers because of internal rules that are not explicitly included in their SLAs, derived for the need to adapt execution to customer needs. We focus on the first situation with public terms in the SLA.The configuration can affect directly to the business control flow (e.g. in our example, if the Documenting Task is always required, the decision taking gateway is omitted), impact on the resources allocation (e.g.: increasing the computation resources for a VIP customer) or modify the execution of a service task (e.g.: configure a service endpoint depending on the customer). The service configuration can be considered either in the *deployment* either in the *fulfillment* phase.

Furthermore, as BPs for different SLAs can be reused, it is required that the BPMS properly **correlate** each BP instance with their corresponding SLAs in the fulfillment phase.

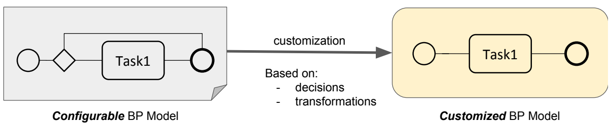


Figure 5.9: Customizing a BP

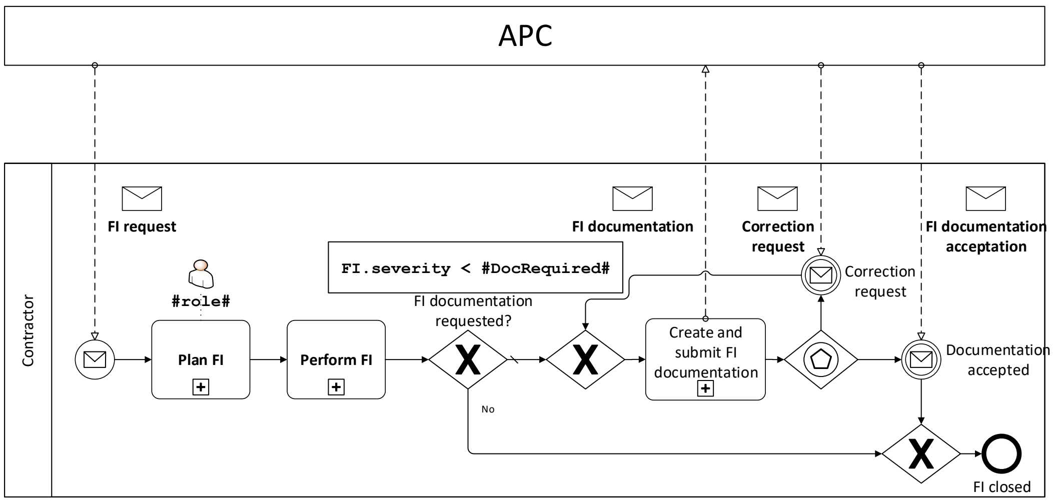
### 5.3.1 Modelling configurable BPs based on SLAs

In previous chapter, we introduced how to extend iAgree with BPMN and PPINOT to specify an SLA for BPs. However, variability or configuration aspects were not considered, so we extend the former specification to support the definition of SLA templates for business processes. The template has to support the configuration information described in the scenario, that is: (i) control flow variations, (ii) process data, (oii) resources assignment or allocation, (iv) configuration data. These configuration options are precisely what the configurable processes provide [155].

There are a number of proposals to define configurable business processes, that can be considered for this purpose [12], [76], [147], [155]. The common approaches to configure process are four: (i) identify configurable nodes (i.e.: tasks) that are optional or can be modified in the configured process [183], (ii) annotate with predicates some elements in the process so the evaluation of these predicates configures the process [12], (iii) specialise some elements in the process so these elements can be instantiated for their related specialised fragment [164] and (iv) describe operations that can be performed over the activities in the process to get the configured version (e.g.: delete a task) [84]. In all these approaches, there is configurable model for BPs and a procedure to configure the process using decisions and transformations (depicted in the Figure §5.9). In our proposal, the configurable BP is included in the SLA template with a simple notation proposed in the following. The decisions for the configuration options are described with pair attribute/values as service terms in the specific SLA, which can be named simply as the *configuration*.

In the Figure §5.10, an excerpt of the SLA template for our example extends the context of the proposed model in section §4.4, introducing two added configuration variables. First, a variable *#DocRequired* (line 8), which can be initiated with a value to constraint the minimum severity of the tasks that have to be mandatory documented. The configuration can include a default value, in case is not configured in a specific SLA. In the example template, the default value is *none*, which means that unless the SLA configs a more demanding criteria, no task is required to be documented. And, as the human resources responsible to perform the task **Plan FI** could be required to have a specific role, such as manager, we have a second configuration variable to indicate it. This variable *role* refers the property *responsible* of the task PlanFI, meaning that the human resource assigned to perform **Plan FI** is required to have the role indicated by this variable. By default, the role indicated in the SLA template is *manager*, but it can be different in the corresponding SLA for each contractor. Furthermore, the process defines different communication tasks with the contractors: (i) *FI request*, (ii) *Correction*

```
id: FI_Service_SLA
version: '1.0'
type: agreement template
context:
   process:
```



```
   configurations:
     DocRequired: [default:none]
     PlanFI.responsible: [#role#; default:manager]
     Timetable: Office
     Calendar: Local
   definitions:
     schemas
       DocRequired: ...
       Timetable: ...
       Calendar: ...
terms:
  metrics:
         ...
```

Figure 5.10: Excerpt of the FI service SLA Template in *iAgree*

*required* and (iii) *Documentation accepted*. The specific endpoints for each contractor (url, communication protocol, etc.) have to be configured in order to enact the process.

In the Appendices, section §C.2, we list the complete example of the proposed SLA Template together with the related metrics referred in previous section and the example of configure SLA.

## 5.3.2 Configuring BPs based on SLAs

In order to apply the described configuration process of two different parties and SLAs to the underpinning processes, two approaches can be followed, which are depicted in Figure §5.11.

On the one hand, each party can use its own deployed and configured process models. This case is depicted in Figure §5.11 **(b) Design-time adaptation**. When a new party signs a new SLA based on the SLA template (Steps 1 and 2), its business

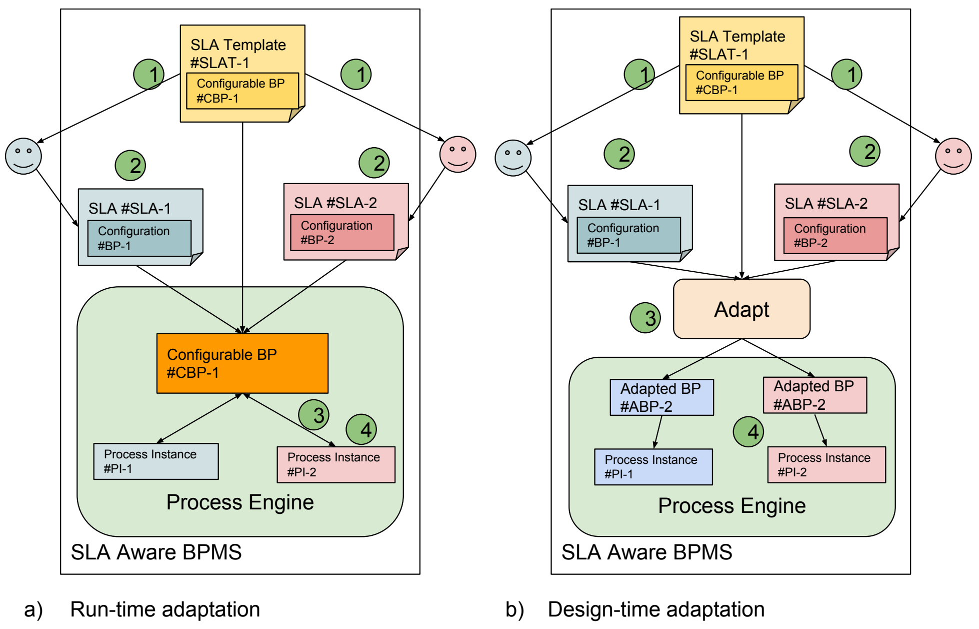


Figure 5.11: Creating Configured BPs

processes are configured for each SLA (Step 3). As these models are exclusive for each party, when a new service instance is required, the BPMS only needs to create a new process instance (Step 4) as no correlation are required. Although the deployed processes are independent for each party, the BPMS could include common information in the models, to ease monitoring or optimise resource management.

On the other hand, different parties can share the deployed configurable process models included in the SLA template, which is described in Figure §5.11 **(a) Run-time adaptation**. First, each customer checks the SLA template (Step 1) to propose a SLA with specific configuration based on it (Step 2). This SLA is related to the common deployed business processes for all the SLAs of this Template. As the deployed processes are shared by all the related SLAs, it is required to configure and correlate the process instances on runtime. Therefore, when a party starts a new service instance (Step 3), the process instance is created taking into account the related SLA configuration. In our scenario, in the processes instances for the Contractor B (Table §5.6), the decision variable DocRequired values #Low means that all the issues have to be documented. This value can be provided in the instantiation of the new process instance (Step 3) or postponed until the decision has to be taken. This mechanism configures service instances in runtime and requires handling the correlation of service instances and their SLAs.

Summarising, we identify two different mechanisms to manage the set of BPs from different SLAs.

- Handle the different set of BPs agreed in each SLA (although based on the same original BP) as different BP models. This configuration mechanism is performed in design time (e.g.: no decision gateway for a specific Contractor).

- Handle a single set of BP for different SLAs which requires providing a mechanism in order to the process instances can query SLA terms (e.g.: Checking if Documenting Task is required) and configure execution.

For the sake of simplicity, we apply the first approach, design-time adaptation, but both mechanisms can be applied depending on the case. Configuring and deploying different sets of BPs for different SLAs simplifies correlation but hinders that process engine applies techniques to optimise resources for all the instances of the same set of BPs (that is one of the proposed advantages of a SLA-aware BPMS), as it handles each set of BPs independently.

#### Correlating instances

As effect of deploying the same BPs for different parties (run-time adaptation), any technique (e.g.: allocation planning) applied to share resources will increase its impact but it is required to enforce the relation between specific process instances and their parties as the process model are reused. Therefore, we need to add explicit information in the instances about their corresponding SLA. A simple mechanism would be to include the SLA identification when a process instance is created so when this instance requires to access SLA terms, it can identify the corresponding SLA. However, correlation is not only required to query SLA terms in runtime, but also to provide any mechanism to interact with the specific process instances of a specific SLA. If the correlation is just managed through process instances variables, the opposite operation, that is, identifying the process instances related to a specific SLA would require selecting all the process instances related to the SLAs sharing deployed BP and filtering the process instances by the SLA identification process variable. This processing can decrease the process engine performance so additional mechanisms to correlate process instances and SLA could be considered (such as an explicit relationship map). These correlations will be further discussed in the next subsection.

### 5.3.3 Architecture

In this subsection we propose an architecture to handle configuration extending a BPMS to enhance it with SLA-aware capabilities. Our proposal for monitoring is not linked to use a BPMS because the processing of performance indicators relies on the events related to them, which can be generated from the existing information. However, the management of BPMN requires a tool which is aware of these models, such as BPMSs. The proposed architecture decouples the responsibilities to fulfill the aforementioned requirements described in this chapter section into different components. We also describe the necessary interface required in order to the involved parties, contractor and customer, manage the full BP lifecycle (no matter who provides the process execution supporting system).

Figure §5.12 depicts the components in architecture to manage the artifacts reviewed in the previous section. We integrate the monitoring components (grey coloured) described in Section §5.2 to provide the full solution. The additional components and their responsibilities in this architecture are:

**BP Manager**. It is responsible for managing and validating Business Process Models, including abstract or executable ones (and not in the extended BPMS). The handling of BP models in the BPMS is wrapped by this component.

**SLA Manager**. This component manages Template and SLA documents **modelled** together with their related process models. It is responsible for storing agreement documents, validating them and managing the business process related to them, that is through the BP Manager.

**BP Configurator**. This component handles the **configuration** of the BPs included in the public SLAs with their corresponding BP artifacts: configured deployed BPs and BP instances.This component would have to be extend to support complex configuration and correlation on run-time.

The system provider can handle their SLA templates together with their configurable and/or executable processes using a private API. A second public API is provided so external parties (customer or provider, depending on the responsible for providing the supporting system) can interact with the service. The interface for such API is

In the next listing, we summarize the main operations to manage SLA of the public API.

```
/sla                            -- Operations related to SLAs
POST      /newsla/()/           -- Add an SLA to the system
DELETE    /{$ID}/               -- Cancel an SLA with given ID
PUT       /{$ID}/               -- Update an SLA with given ID
GET       /{$ID}/start          -- Starts a new instance related
                                     to the SLA given by ID
POST      /{$ID}/{$PID}/{}      -- Provide a message to the process
                                     instance PID of the SLA given by ID
GET       /{$ID}/evaluation -- Retrieve the evaluation of the SLA
                                     given by ID
```

Document 5.1: Java code of the getAgreement operation

## 5.3.4 Applicability of our proposal

We have developed a prototype with the proposed components and interfaces as an extension to an open source BPMS, Camunda[4]. We describe some of the implementation decisions. Camunda provides two different interfaces to interact with processes.

[4]https://camunda.org/

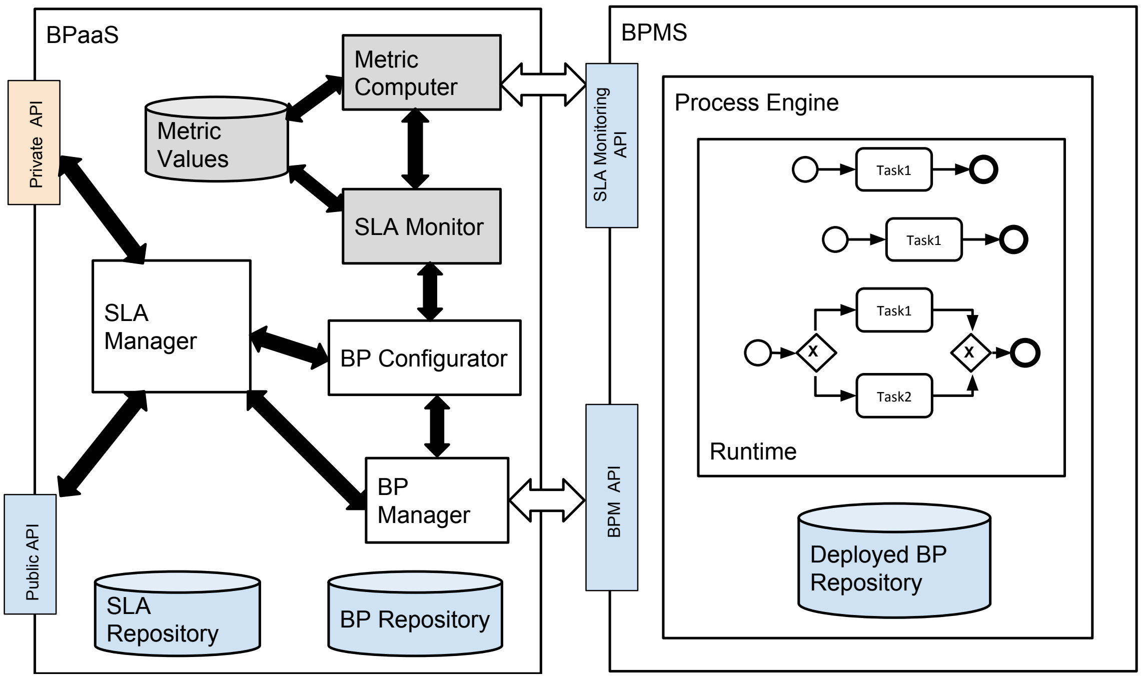


Figure 5.12: Architecture

A Java interfaces and a REST API provides operations to deploy business models, start new process instances or monitor instances runtime information. Although our architecture would increases decoupling using the REST API, for network performance reasons, we have developed the prototype with the JAVA service interface.

As we use BPMN to describe BPs our SLA documents include these XML models. The BP models are stored in BP repository and managed by SLA manager. As the provider can define business processes before relating it to an SLA or template, BP documents are managed by BP Manager. As common BPMSs have their own storage system for deployed BPs, when SLA manager requests the deployment of an executable BP in the BPMS, it also registers the identification of the deployed BP model to correlate SLA and deployed BP model. This correlation is required to initiate, handle or monitor service process instances. Therefore, the BPMS is decoupled of the SLA management but as it is completely unaware of it, if the BPMS updates model identification, the correlation between SLAs and deployed BP models would be broken. To avoid this, we could include mechanisms to keep correlation in the BPMS storage. A simple mechanism is using an universal identifier for deployed BP model which includes the related SLA identification, such as:

```
#deployedBPID = #BPID@#slaID
```

where *#slaID* is the unique identifier of the SLA and *#BPID*, the unique identifier of each business process included in the SLA.

Doing so, we also avoid some interaction with BPMS, i.e. querying the executable process model related to an SLA template in order to include it in a new SLA. An executable BP commonly includes other resources apart of BPMN documents, therefore the executable BP related to an SLA includes not only BPMN documents but any other information required for deployment such a code sources files (e.g. .java files in Camunda BPMS) or images (BP model).

In the previous subsection §5.3.2, we described different choices to configure business process according to the SLA. In our developed prototype, each configuration parameter, such as *DocRequired*, is mapped as a process variable and its values are included in the deployed process model using the design-time approach. This mechanism is only valid if the specific SLA includes the specific configuration value. If this value requires run-time processing through the SLA lifecycle, additional mechanisms should be provided. In our running example, in order to be consistent with this approach, the decision gateway related to execute or not the documenting task has to be based on this process variable, *DocRequired*.

A full example of the use and responsibilities of these components is described in Figure §5.13, where an example of the main activities in the SLA lifecycle are depicted.

In **Step 0** in Figure §5.13, the service provider defines a SLA template, #T1, which includes a simplified version of the example process, #CBP1, with a Task and decision gateway, and a guarantee term, DocTime lesser than 4 hours. This process is defined with BPMN and this metric, DocTime, as the time difference between the process starts and the Task is finished. The decision gateway depends on the configuration option, DocRequired. The provider uses the **Private API** to register the template. Then the API uses the **SLA manager** to check and register the SLA template (Step 0.1). The **SLA Manager** extracts the business process and requests to **BP Manager** to evaluate and register it (Step 0.2). As it is a configurable processes is not deployed in the process engine. Once the SLA template and related processes are registered, potential customers can query them with a public API to analyse and negotiate possible SLAs (**Step 1** in Figure §5.13).

When a customer signs a new SLA, it is registered with the public API. In the **Step 2** in the Figure §5.13, a new SLA, #S1 is proposed based on the SLA Template #T1, where the DocRequired has the value of all, which means that the Task is always executed, and a guarantee term with a similar metric to #T1, DocTime, but more restrictive target value, 2 hours. This SLA is managed by **SLA Manager** (Step 2.1) and then processed by BP Configurator. This component checks the SLA document and included process and requests to the **BP Configurator** (Step 2.2) to evaluate the SLA terms and the process to propose a deployable process. In the example, the process included in the SLA is similar to the SLA Template but, as the Task is always execute, the decision gateway is not required. As the process model in the template could be reused in different SLAs, **BP Configurator** defines a configured process, #ABP1, which is handled by **BP Manager** (Step 2.3), component and deployed in the Process Engine, through its **BPM API** (Step 2.4).

In a similar way, when a new service execution related to SLA #S1 is requested through public API in the **Step 3**, the request is received by **SLA Manager** (Step 3.1). This component uses **BP Configurator** to properly correlate with the BP model (Step 3.2) to be executed. In the example, the term Required Task in #S1 has to be included as a variable in the process execution so the decision gateway included in the deployed BP is properly evaluated. Then, **BP Configurator** requests to **BP Manager** to start a new process instance of #ABP1 (Step 3.3) and this component uses the **BPM API** to initiate the process instance (Step 3.4). The identification of the process instances has to be managed by the BP and SLA Manager to correlate the SLA with its process instances. We have to highlight that, in order to simplify the scenario, we log all the process instances events so any performance information can be queried in any future moment. In a real environment where resources overuse can decrease performance, the metrics included in #S1 should be considered to only log the relevant events for its metrics (process start and end of Task events).

Although it was covered in previous section, we also include the corresponding SLA evaluation stage in this diagram. When it is requested to evaluate the SLA #S1 accomplishment through the **Public API** in the **Step 4**, this request is handled by **SLA Manager** (Step 4.1), which uses **BP Configurator** to correlate the SLA terms with the deployed BP model and instances (Step 4.2). And this component asks the **SLA Monitor** to evaluate the SLA terms, such as DocTime < 2h (Step 4.3). To evaluate this metric, it is required to compute the value of DocTime, what it is performed using the logged events and the **Metric Computer** (Step 4.4). The result of the evaluation is returned through the **Public API**.

In our running example, the initial service template is proposed by APC, the service customer, so the party responsible to review template and propose a specific SLAs and configured business process are the potential contractors. In a BPaaS scenario, the responsible to define the template is the service provider and the potential customers are responsible to propose their configured SLA.

### 5.3.5 Related Work

Current BPMSs provide mechanisms to support process instances related to different parties, but their support for SLA is limited. IBM Process Standard [5] provides a notification system based on SLAs. These SLAs are sets of performance indicators over the process instances. Notification helps to take preemptive actions or decisions on runtime but these actions are manual and automatic actions has to be made as an extension to this engine. There is no mechanism to automatically customise process execution based on their so-called SLAs.

Some research proposals also address the configuration of executable business processes. Gottschalk *et al.* [76] and van der Aalst *et al.* [183] propose a mechanism to

[5]http://www-03.ibm.com/software/products/business-process-manager-family

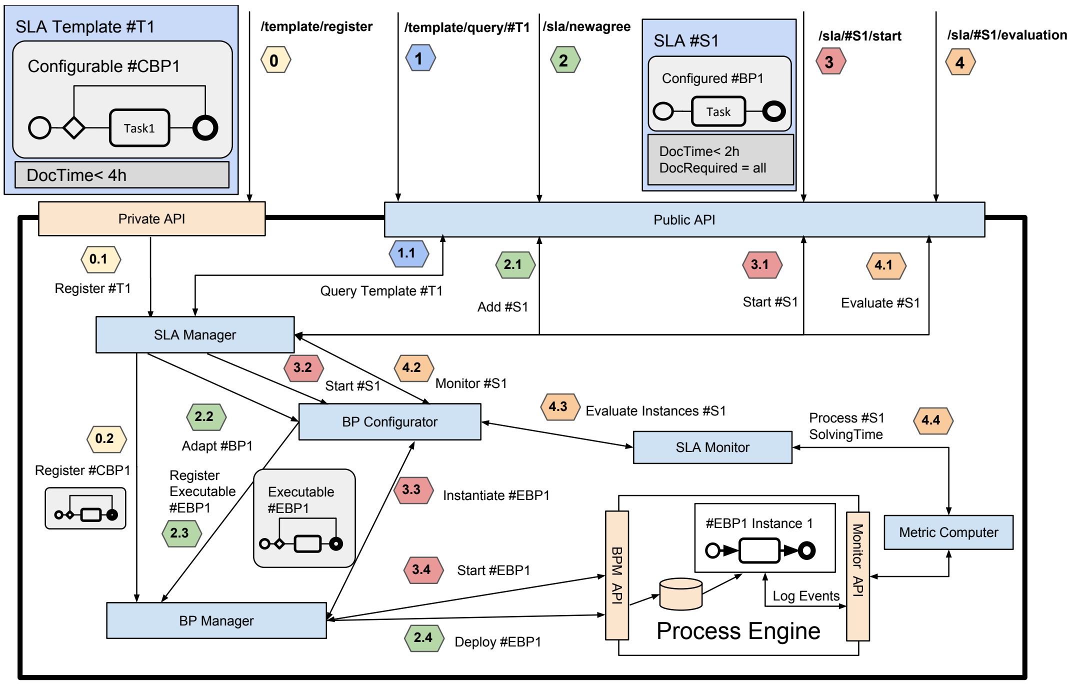


Figure 5.13: Use of Management and Monitoring APIs

define configurable business processes or workflows which can be configured for each node based on different attributes on design or execution time. They do not consider named SLA with SLOs. Kumar and Yao [102] use logic language to describe rules which customize the process instances before executing them on a workflow engine. Their goal is provide a robust process design that makes it easier to accommodate changes to business policy. And, lastly, Acher *et al.* [5] use feature models to describe a family of services in such a way that allows reasoning about the compatibility between connected services to ensure their consistency and propagating the variability choices when configuring services. All these proposals focus on providing configuration mechanisms but they are based on the SLA so we could extend our proposal to include them.

## 5.4 Summary

In this chapter, we address the management of lifecycle of SLA for BPs, introduced in the previous chapter. Regarding the late stages in SLA lifecycle (Deployment and Fulfillment), we identify three operations of interest: monitoring, configuration and optimization, although the latter is out of the scope of this dissertation. With respect to monitoring, after analysing the different approaches to process performance data we provide a reference component to monitor SLA accomplishment which is independent of the supporting PAIS. This proposal enables the run-time evaluation in different

PAISs of performance indicators defined and has been applied to a real monitoring scenario. Furthermore, some mechanisms are proposed to increase efficiency.

Regarding the configuration of BPs for different parties, we provide a mechanism to support the creation of configured BPs based on an SLA template and specific configuration for each party. These configured BPs are deployed and executed corresponding to their related SLA. The application of the proposal is developed through an architecture to extend an existing BPMS and it has been validated through the implementation for an open source BPMS, Camunda. In order to properly validate that this proposal is independent of the supporting BPMS, it should be extended to a wider range of systems. This effort shall be carried out in future work.

# Frame Agreements

*This is an era of specialists, each of whom sees his own problem and is unaware of or intolerant of the larger frame into which it fits.*

*Rachel Carson ( 1909 - 1965),*

## 6.1 Introduction

Certain scenarios of outsourcing of non computational services, such as IT development projects or transport&logistic services (T&L), are ruled by a frame agreement for a time period or for a budget limit. This frame agreement defines the characteristics of the services provided and some commitments between the parties within that period. However, in contrast with the scenarios previously analysed, in this context each service instance has its specific agreement that regulates the aspects that are specific to it.

This is the case of a new transportation request in the T&L context, where large transport companies can operate 100.000 transportations in a random month or IT development contracts where each work order has its own agreed resolution times. These frame agreements are usually described with natural language, so supporting their evaluation requires developing ad-hoc mechanisms.

In this dissertation, we provide a model to formalise frame agreements supported by process-aware-information-systems (PAIS) so the global commitments or the ones related to single actions or service instances can be automatically monitored. We had previously analysed in [81] the conformance between aggregated and specific metrics in terms of constraint programming. In this chapter, we reinterpret the problem in terms of an SLA for Business Processes.

The contribution of this chapter regarding the dissertation is highlighted in Figure §6.1.

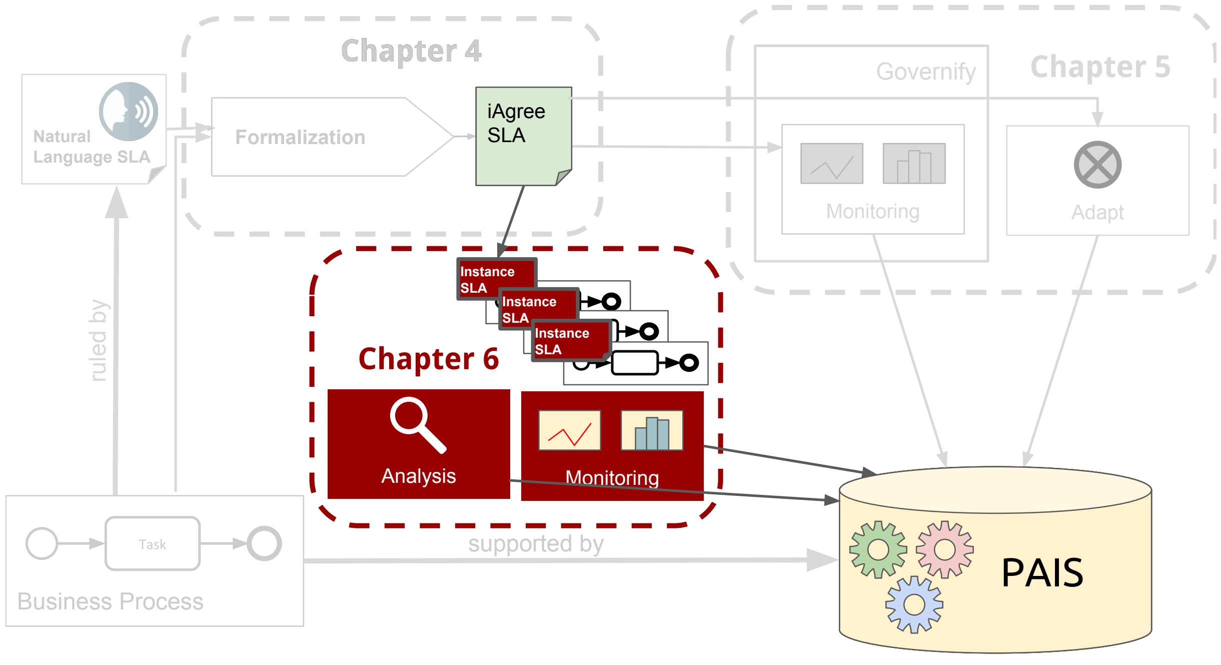


Figure 6.1: Contributions background

## 6.2 An example scenario

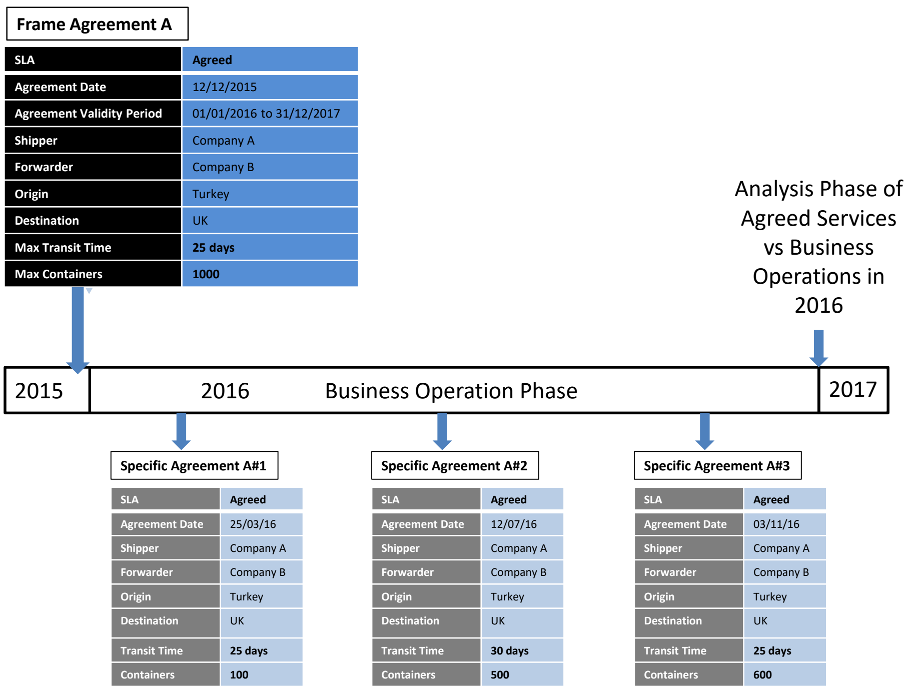
We identify here two different scenarios where the frame agreements are described with different scopes. On the one side, when transport & logistics enterprises outsource transport services, it is common that they sign an agreement for a long validity period, such as a natural year, although each specific transport provision has different execution terms. On the other side, in IT development contracts, companies usually outsources the development of a component given a limited budget but each iteration on the development of the component is managed with a single work order with their own performance requirements. In this section, we describe these two scenarios.

### 6.2.1 Service Agreements in Transport and Logistics

In the case of agreements in transport & logistics between international parties, a forwarder, which regularly sends good from Asia to Europe, contracts a shipper for annual periods so they are guaranteed that certain terms are held for the whole year, such as price per container, although they specify a new specific agreement per shipment where they include specific conditions for that shipment such as dates for delivery. These frame agreements usually define other limitations to their scope apart from validity period, such as maximum number of shipments of appliance or container contents (e.g.: not biologic contents) so the pricing per container is constant for the considered frame agreement. The evaluation that the agreement for a single shipment or the shipment itself is consistent with the frame agreement, i.e. the required number of containers to be shipped does not exceed the scope defined in the frame agreement, is manually performed at the end of the agreement so preemptive actions cannot be taken if a single shipment violates the frame agreement and the checking of consistency for the whole year is a tedious, error-prone task. This scenario has already been analysed with an approach to handle agreement conformance in our proposal in [81].

The scenario described in that work is based on real life situations described by domain partners of the European FInest project [114] and it is illustrated in Figure §6.2. A shipper establishes in the end of 2011 a frame agreement #A with a forwarder. This frame agreement defines that the forwarder will organize the shipments of the goods with a certain set of SLOs. In this example, the SLO *maximal transit time* is defined with value *25 days*, as well as the SLO *max containers* which determines that maximum 1000 containers could be transported for the same agreed price during the validity period of frame agreement #A. For each execution of a transport & logistics process associated with the frame agreement #A, a specific agreement is created. In this example, three specific agreements were established during the year of 2012 for each time the shipper requested a service. One request at the beginning of 2012 originates the specific agreement A#1. A second request originates the specific agreement A#2 and another one creates the specific agreement A#3. At the beginning of 2013, both shipper and forwarder analyze their business operations during the past year to identify eventual violations.

In the case illustrated in Figure §6.2, one violation happened. According to the frame agreement A, only 1000 containers could be transported under this agreement. However, at the time the specific agreement A#3 was created, the cumulative transported cargo under A was 550 containers (the aggregated value of the SLOs called *Containers* in specific agreements A#1 and A#2). The remaining amount of allowed cargo was 450 containers, but the shipper transported 600 containers as described in the specific agreement A#3. Thus, the shipper violated the frame agreement A by having extra 150 containers transported for the same price of the allowed 450. Penalties upon this partner will be enforced very late after the actual losses in the perspective of the forwarder were suffered. This happens because the conformance checking in transport & logistics agreements is currently a manual process executed typically every 6 or 12 months [114]. Such manual process could be viable in a small company, but in large companies with high volumes of specific agreements such manual processes becomes very costly. Hence, new online, automated conformance checking mechanisms can drastically improve the violation detection in transport & logistics SLAs and this could lead to cost reductions.


Analysis Phase of Agreed Services vs Business Operations in 2016
2015
2016
Business Operation Phase
2017

Frame Agreement A

| | |
|---|---|
| SLA | Agreed |
| Agreement Date | 12/12/2015 |
| Agreement Validity Period | 01/01/2016 to 31/12/2017 |
| Shipper | Company A |
| Forwarder | Company B |
| Origin | Turkey |
| Destination | UK |
| Max Transit Time | 25 days |
| Max Containers | 1000 |

Specific Agreement A#1

| | |
|---|---|
| SLA | Agreed |
| Agreement Date | 25/03/16 |
| Shipper | Company A |
| Forwarder | Company B |
| Origin | Turkey |
| Destination | UK |
| Transit Time | 25 days |
| Containers | 100 |

Specific Agreement A#2

| | |
|---|---|
| SLA | Agreed |
| Agreement Date | 12/07/16 |
| Shipper | Company A |
| Forwarder | Company B |
| Origin | Turkey |
| Destination | UK |
| Transit Time | 30 days |
| Containers | 500 |

Specific Agreement A#3

| | |
|---|---|
| SLA | Agreed |
| Agreement Date | 03/11/16 |
| Shipper | Company A |
| Forwarder | Company B |
| Origin | Turkey |
| Destination | UK |
| Transit Time | 25 days |
| Containers | 600 |

Figure 6.2: Frame agreement for transport & logistics

### 6.2.2 Service Agreements in IT Development

Similar to annual contracting in transport & logistics, we find the outsourcing of development services, where an agreement is signed with the development company at the beginning of the project. In this agreement, they define terms such as deadlines for development tasks or the maximum total budget. Each time a new work order (WO) is requested, the service provider plans and estimates specific terms for this work order, such as expected finishing times or estimated budget for this work order. These terms for a single work order should take into account the agreement because the maximum budget could be insufficient so the work order should be cancelled or redefined. When agreement is signed, some terms specific to each single work order could also be specified, such as the maximum time to solve each one. This global restriction could be mandatory watched in the each service execution or they could be just a default term (e.g.: $maximumsolvingtime < 2d$) to be redefined in a new work order, as more restrictive constraint (e.g. $maximumsolvingtime < 1d$), more relaxed constrained ($maximumsolvingtime < 3d$) or even discarding the constraint.

The abstract process for work orders is depicted in Figure §6.3. When the validity period of the agreement starts, new work orders must follow the defined BP. When a new work order is requested, the provider *plans* the work effort for this work order. The result of the plan is a specific agreement just for this request and includes the estimated cost or tasks deadlines for this work order. When the plan is approved, the provider *develops* and then *documents* the work order. Lastly, the work order is *delivered*

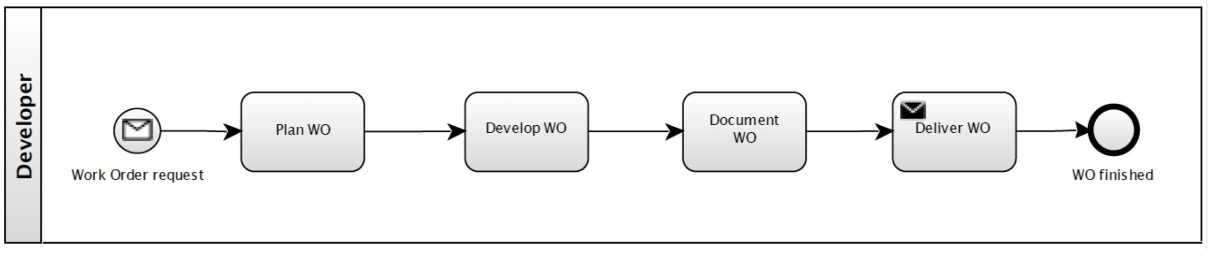


Figure 6.3: Business process for work orders

into the customer infrastructure.

In Figure §6.4 there is an example of status for outsourcing with different instances of work orders. Before starting the IT outsourcing, the signed agreement #A defines a budget of 1000 € for the outsourcing and constraints that all the requested work orders have to be developed in a maximum of 3 days. A first work order #1 was requested, planned (with an estimated cost of 200 €) and developed in 2 days so it is consistent with the agreement. A second work order #2 was requested, planned (with an estimated cost of 400 €) but development took 4 days to finish so this execution it is violating the agreement. A third work order #3 was requested and planned (with an estimated cost of 300 €and it is currently in development). And a fourth work order #4 has been requested but it has been planned with an estimated cost of 400 €. Although these four work orders have not been finished, their total estimated costs sum 1200 €, 200 € over the planned budget, so the plan for the work order #4 cannot be approved. Therefore, a decision over the execution has to be taken, as cancel or replan the work order #4. The process could consider review estimated costs in any stage of the rest of work orders and the real final cost should be considered when the work order process is finished. Furthermore, when a new work order is planned, the required *Deliver Date* is specified as a deadline for the delivery of the developed work order. The *delivery date* is a guarantee included in each specific agreement and each one is independent of the rest of the specific agreements. If there were a general constraint for every work order (e.g.: the time from starting the work order to the delivery has a common limit for every work order), it could be included in the frame agreement.

## 6.3 Frame Agreements

In this section, we are going to analyse how to model the SLAs for the introduced scenarios using the model proposed in the previous chapters for non computational services. As in these scenarios two level of SLAs appear, we also evaluate the impact of their relationships in the SLA lifecycle and how to address the operations related to the different phases of the SLA lifecycle.

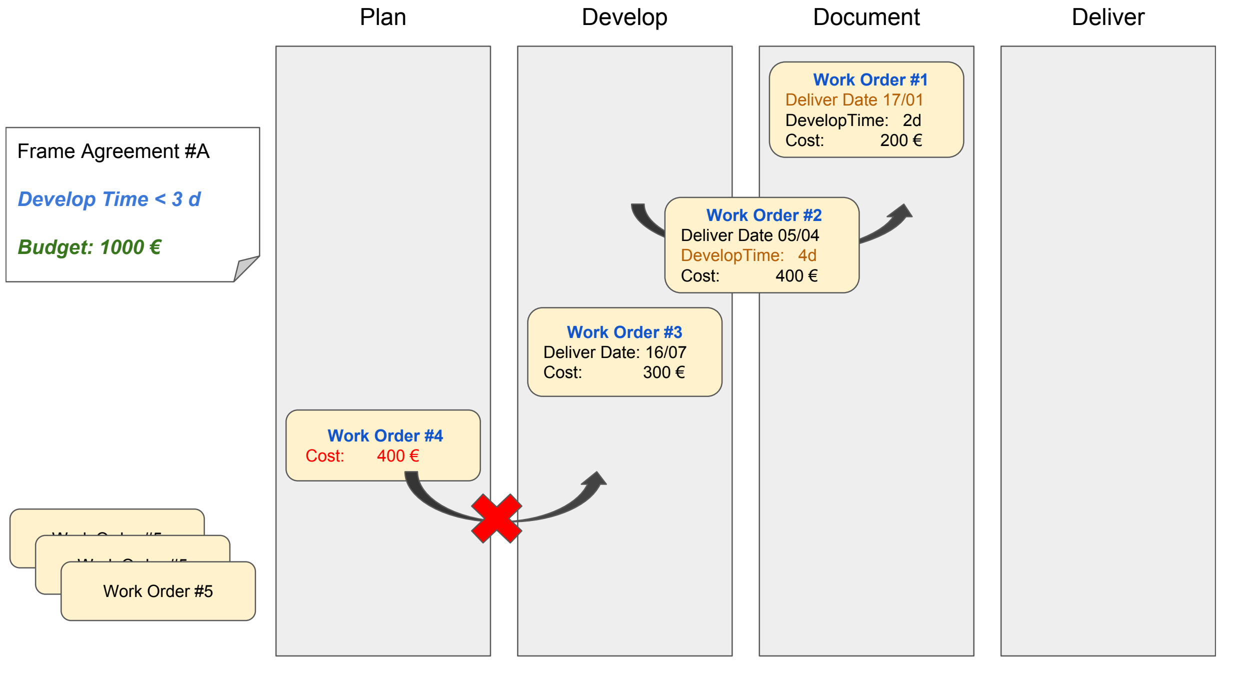


Figure 6.4: Example of SLA Hierarchy for work orders

### 6.3.1 A conceptual model for frame agreements

The example scenario introduced in the previous section (Figure §6.4) includes a cascade of commitments between parties which exchange a service (usually, consumer and provider). We establish the relationship between a frame agreement (FA), and a specific agreement (SA), in three points, that are depicted in the conceptual model in Figure §6.6. First, the traceability between a FA and its SAs (relation between Agreements). Second, a FA can include general requirements for all the service instances (e.g.: *Develop Time* < *3d*), so they will be included in each related SA. These are atomic terms. And last, aggregated metrics in the FA (e.g.: *Budget* = 2000€) should be related to the corresponding metric in the SA (e.g.: instance cost = 500€). In more complex scenarios, a specific agreement can also be frame agreement of more specific actions in its scope. For example, the development activity could be composed of different subtasks: (i) purchase required equipment, (ii) planned onsite implementation and (iii) later remote verification; and each of these subtasks could have a more specific agreement ruled by the SLA of the work order, which acts as its frame agreement.

Recursively, more specific agreements could describe subprocess and extend the SLA hierarchy, as it is depicted in Figure §6.5, where three layers of SLAs are displayed. Within the context of a frame agreement, A, such as the SLA for IT outsourcing, more specific agreements, A1, A2 and A3, can appear, one for each kind of tasks in the IT outsourcing: development, maintenance and IT training, and each kind of tasks can have more specific agreements for each task instance (e.g.: a specific training period),

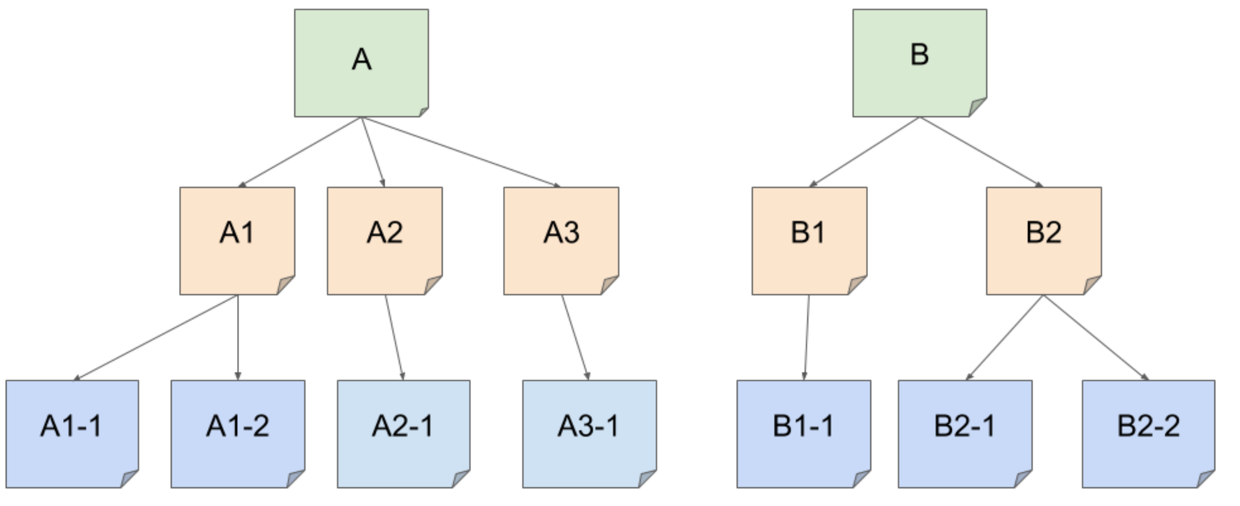


Figure 6.5: SLA Hierarchy

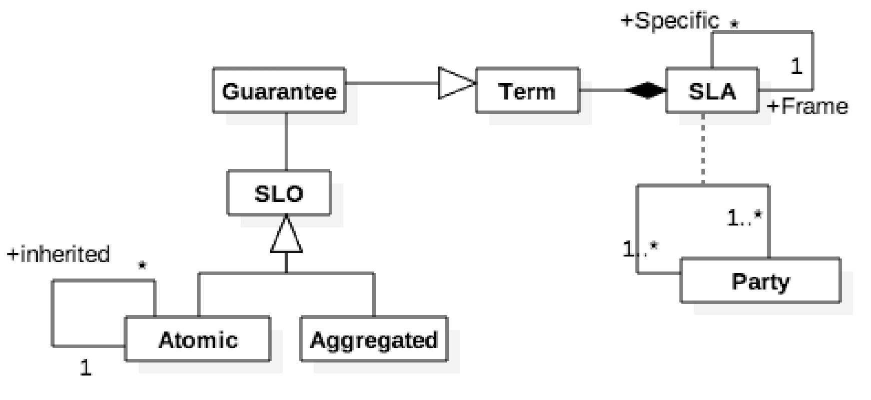


Figure 6.6: Conceptual Model for frame agreements

A1-1, A1-2, A2-1 and A3-1.

### 6.3.2 Lifecycle of the agreements

Now, we review the interaction between FA and SA and the required operations to properly manage them. As the specific agreements are related to specific service instances in the context of the frame agreement, they are only created when the frame agreement is already created and deployed, that is, during the *fulfillment* phase of the frame agreement. The lifecycle of a specific agreement is depicted over the SLA Lifecycle, which was introduced in Chapter §3, in the Figure §6.7. First, the creation of a specific agreement is one of the first activities in a new service instance (e.g.: the first activity of a work order is planning it). It is *created* according to the atomic inherited guarantees from its frame agreement. After its creation is on the pending state. If there is *conformance* between frame and specific agreement (e.g.: the available Budget in the frame agreement has to be greater than the estimated cost for a work order), the spe-

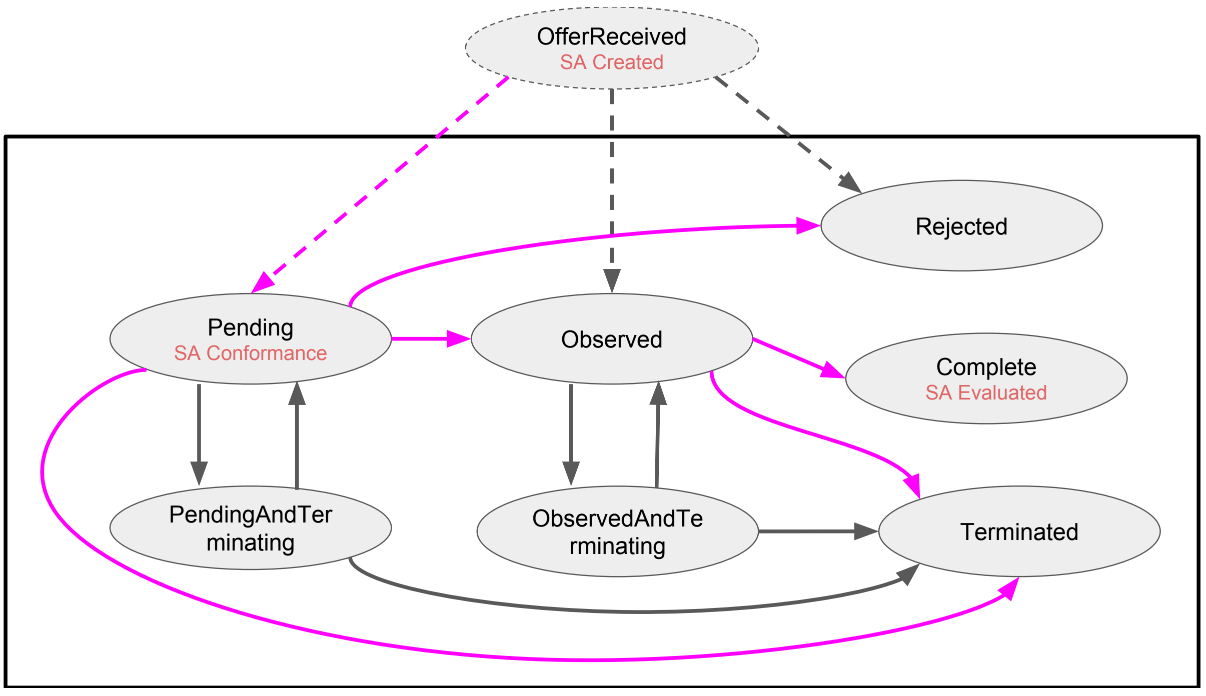


Figure 6.7: Agreement states in WS-Agreement

cific agreement is considered valid and it is in the *Observed* phase. During this phase we observe the specific agreement fulfillment. And, finally, the executed terms have to be *evaluated* not only regarding specific agreement but also with frame agreement (e.g.: the final Cost can be higher than the estimated Cost, so the available Budget after the work order execution would decrease). Therefore, we identify three different operations to support SA Lifecycle: (i) creation of new specific agreement based on a frame agreement, (ii) conformance between specific and frame agreement, (iii) evaluation of execution regarding specific and frame agreement.

## 6.4 Operations

In this section we are going to detail the operations defined in the previous one. A simple example for each operation is depicted in the Figure §6.8.

### 6.4.1 Creation of a specific agreement

When a new service instance is created in the context of a frame agreement, a new specific agreement has to be defined. The terms in frame agreement that impact single service instances are inherited in the specific agreement. The Figure §6.8 (1) depicts the creation of a new specific agreement *S#1*, related to the frame agreement *A#1* and which inherits the goal *Develop Time* < *2d*.

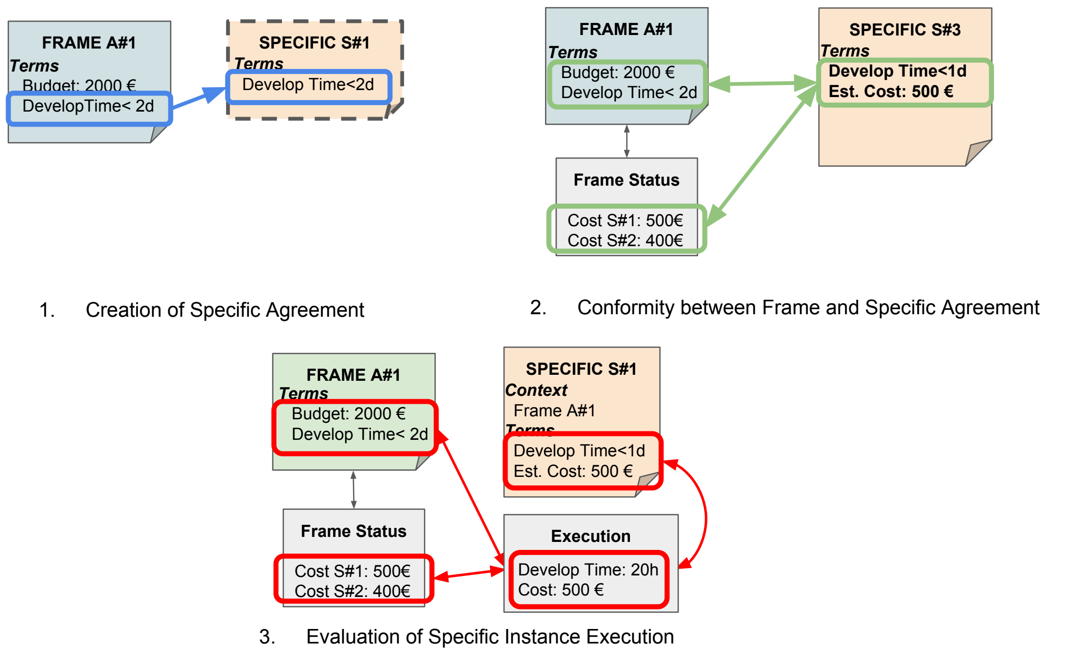


Figure 6.8: Examples of Operations between frame and specific agreements

As a result of this operation, a new specific agreement is created referring to a frame agreement.

### 6.4.2 Conformance between frame and specific agreements

Once the specific agreement is completely defined, the conformance between the frame and the specific agreement can be analysed, not only including inherited terms but also other terms related to aggregated terms (e.g.: The *Estimated Cost* that has to be considered for the aggregated metric *Budget*). We consider that a frame and specific agreements are conform if:

- (i) In atomic guarantees that are related between specific and frame Agreement, the former implies the latter. That is, the SLO in the specific agreement has to be, at least, so restrictive as the SLO included in the frame agreement. For example, as depicted in Figure §6.8 (2), if the frame agreement defines a guarantee $DevelopTime \leq 2d$, the specific agreement can define a guarantee $DevelopTime < 2d$ or $DevelopTime \leq 1d$ but not a guarantee $DevelopTime \leq 3d$.
- (ii) Regarding to aggregated SLOs, frame and specific agreement are conform if considering the related metric values for previous specific agreements and the current defined specific agreement, the aggregated SLO is fulfilled (e.g.: If the frame agreement defines an SLO $Budget = 1000$ and two work orders have finished with a cost of 400€ and 300€, respectively, a new work order that esti-

mated a cost of 300€ is not conform, as the sum of finished costs, 400€ + 300€, and the estimated one, 300€, overcomes the budget, 1000€.

A detailed description of conformance and Constraint Satisfaction Problems (CSPs) to solve this operation is included in our previous work [81].

This operation returns a boolean (conform or not conform) and a negative result can determine that the specific agreement is out of the scope of the frame agreement or it is replanned.

In Figure §6.9, we describe in detail the evolution of Budget aggregated value regarding specific agreements (estimated cost) and service execution (real cost).

### 6.4.3 Evaluation of specific instance execution

During service instances, their execution has to be monitored in order to properly evaluate SLAs accomplishment, both its specific and frame agreement. The evaluation should be addressed regardless of the conformance between specific and frame agreement. The expected evaluation is that execution holds both specific and frame guarantees, but it could happen that execution matches specific agreement guarantees and does not match frame agreement guarantees. Fro example, if the frame agreement defines a guarantee $DevelopTime < 4days$ which is even constrained by the specific agreement to $DevelopTime < 2days$, both agreements are conformance. But if the real Develop Time is 3 days, the execution would be not valid in the context of the specific agreement but it would be valid in the context of the frame agreement. Evaluating the execution is similar to the conformance but using specific execution values (e.g. Develop Time = 2 days) instead of the guarantee constraints (e.g. Develop Time < 2 days). In Figure §6.8 (3), there is an example of execution values, with $DevelopTime = 20h$ and $Cost = 500$ €. These values can be directly evaluated against the constraints in specific and frame agreements. In the case of *Develop Time*, it accomplishes both guarantees ($DevelopTime < 2d$ in frame agreement and $DevelopTime < 1d$ in specific agreement). And, once the instance is finished and we have the final *Cost*, the Estimated Cost of that instance is not relevant for the processing of *Budget*.

After evaluating the metric values together with the objective, different SLO statuses can result. If all the SLOs are fulfilled, the specific agreement is considered *terminated* and the frame agreement would be still being *observed*. However, if one of the SLOs is not fulfilled, different situations can happen. On the one side, when the real Develop Time is 3 days, both the atomic SLO $DevelopTime < 2d$ in frame and specific agreement are *violated*. On the other side, if atomic SLO in specific agreement is more restrictive than in frame agreement, it could happen that the atomic SLO in specific agreement is *violated* and the one in the frame agreement is not. In the aggregated metrics, the metric in the service instance is not guaranteed by the specific agreement so the final metric value could unfulfill the aggregated SLO in frame agreement although the specific agreement is fulfilled (e.g.: when the available Budget is 300€and

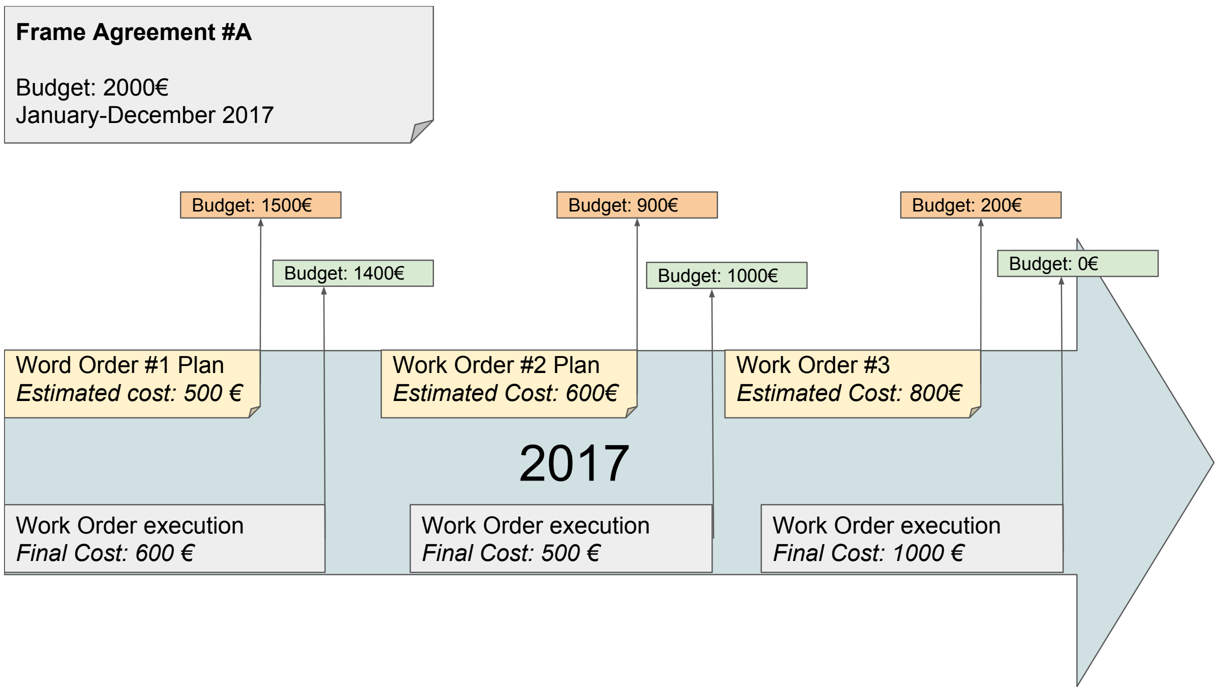


Figure 6.9: Evolution of Metrics

a new work order that estimates a cost of 200€ is approved. If the final cost is 400€, the aggregated SLO would be unfulfilled). In any case, as a result of this evaluation, a manual decision could consider that the specific agreement and the related service instance are out of the scope of the frame agreement, so neither the specific agreement neither the service execution has impact in the frame agreement status (e.g.: if we have an available Budget of 300€ and a new work order costs 400€, it could be considered out of the scope, so the available Budget would still be 300€).

## 6.5 Interpreting Frame Agreements as SLAs for BP

We propose modelling frame and specific agreements with our SLA model for BP, introduced in Chapter §4. First, the services described can be modelled with BPs. Second, atomic metrics and related guarantees can be defined with metrics of base measures in PPINOT. And, last, aggregated metrics can also be defined with aggregated measures in PPINOT. Therefore, we interpret these scenarios as an SLA for Business Processes and we formalise specific and frame agreement with our proposed model. Then, the SLA for a single work order includes only single instance metrics and the frame agreement both atomic guarantees for each single process instances and aggregated metrics for the whole service.

In order to address the identified lacks and support the proposed architecture, one choice is developing an extension to our SLA management tool, Governify [1], to support the frame and specific agreements and the operations proposed in this chapter.

Therefore, given the requirements of our scenarios, where the specific agreements do not redefine the atomic guarantees but just instantiate the metrics, we take an interpretation to use one single SLA level, that we describe in this subsection (e.g.: if the frame agreement defines an SLO of $DevelopTime < 2d$, the specific agreement defines a deadline for the development 2 days later than the start of development as a consequence of the SLO in frame agreement). If we consider the frame agreement as our SLA for BPs and a specific agreement as a single service instance that we need to monitor, the frame agreement is described with our SLA model, and the specific agreements are defined as the events that are monitored using the Infrastructure element in the SLA. The guarantees and performance values of the specific agreements are stated as process data and included in the events. With this interpretation, the applicability is simplified as all the operations are performed over the same agreement but we have to differentiate between the events related to the specific agreements and the real executed events for the service. Therefore, the metrics should have to be described to consider both events. We consider both creating two metrics, one for the estimated metric values and other for the executed metrics values. An example of this interpretation is explained and depicted in next subsection, where a frame agreement is modelled as an SLA for BP and the specific agreements as the monitored events.

The model that we introduced in Chapter §4 only lacks of the expressiveness to explicitly relate the frame and specific agreements and describe the relationship between terms from frame and specific agreements (e.g.: the relation between the atomic guarantee *Develop Time* < *2d* in the frame agreement and the corresponding in the specific agreement, or the relation between aggregated metric, *Budget*, in the frame agreement and the related metrics, *Estimated Cost* or *Final Cost*, in the specific agreement).

### 6.5.1 Materialising the operations

The implementation the aforementioned operations using our interpretation would have the following schema, which is depicted in the Figure §6.10 for our example. First, the frame agreement defines the aggregated metrics as the aggregation of two different atomic metrics: a metric in the specific agreement and a metric for the execution value together with atomic ones (e.g. Develop Time in the Figure §6.10). Then, when a new service instance is created with a new specific agreement, a new event is created with the related data for the atomic metric in the frame SLA, which are inherited. Then, when the work order is planned the specific atomic metrics are defined in a new Event and the conformance between frame and specific agreement is evaluated as monitoring the execution. When the work order finishes, the real value substitutes the estimated one. The conformance and evaluation of execution are performed similar, checking

[1] `http://governify.io`

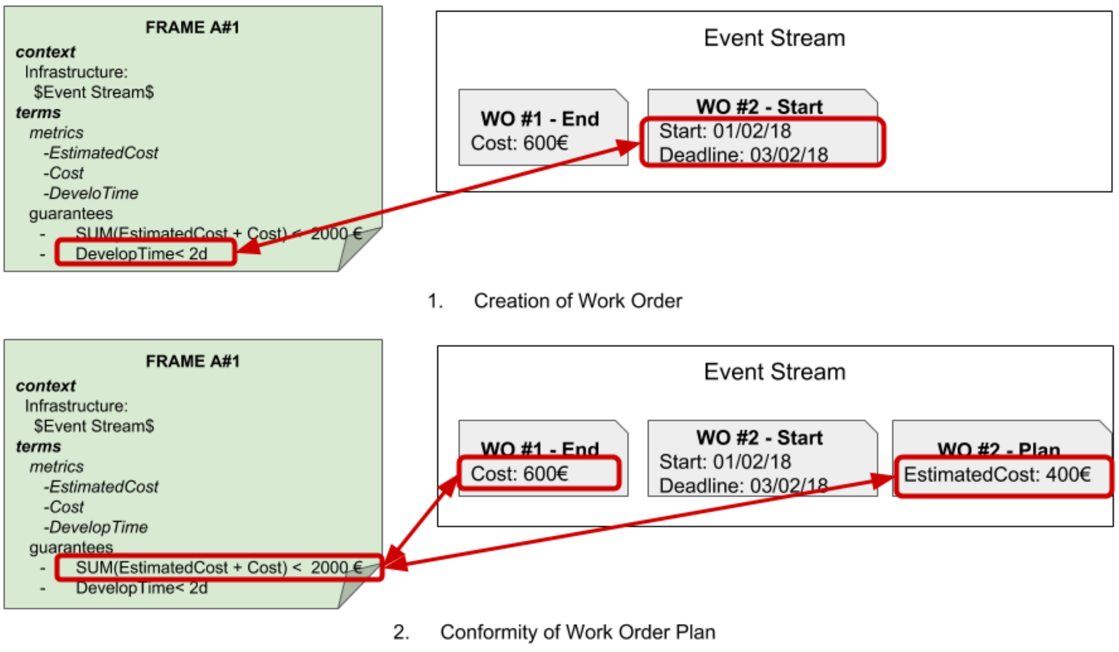


Figure 6.10: Operative with one single document

that the monitored values of both, execution and estimated event, accomplishes the aggregated SLO in the frame agreement.

In the example in the Figure §6.10, a Work Order has a **Develop Time**, which (in order to simplify the explanation) is the total time since the Work Order starts until it finishes, an **Estimated Cost**, which is the planned cost of the Work Order at its beginning in order to manage resources, and a final Cost, which is the real **Cost** when the Work Order has been performed. As it is depicted in the Figure §6.10, the frame agreement limits the Develop Time of all the Work Orders to 2 days, which is an atomic SLO and the total Budget for all the Work Orders to 2000€, which is an aggregated SLO, defined as the sum of Estimated Cost and Cost. In the Figure §6.10 (1), a new Work Order is request, and as it inherits the atomic SLO of 2 days for Develop Time, a new event is created for this Work Order, where the expected Deadline for this new Work Order #2 is 2 days after the start of the Work Order. In the Figure §6.10 (2), the Work Order is planned with an Estimated Cost of 400€and a new event is created. The conformance between the Work Order is checked with the aggregated SLO, Budget, using also the Cost from previous Work Orders. The evaluation at the end of the Work Order is performed similarly, creating a new event and using the final Cost of Work Order #2 instead of the Estimated Cost.

The frame agreement is listed in the Appendices, Section §C.3, together with the related metric computers.

## 6.6 Related Work

There are a number of proposals related to the creation and evaluation of services in domains like transport & logistics or IT Maintenance. However, as far as we know, none of the evaluate the existence of multiple levels of agreements. First, in our previous work, [81], we presented an analytic model to evaluate these multiple levels of agreements, providing a definition and implementation of the conformance between frame and specific agreements based on CSP. Then, in [115], Cassales *et al.* , extended this proposal to provide an SLA Management Framework for Transport and Logistics Services.

In the past years, there is an increasing amount of research efforts specifically trying to bring into the transport & logistics domain technical solutions based on Service Oriented Computing [199], [13]. Nevertheless, when it comes to SLAs management there is a limited amount of work.

The work proposed by Zhu and Fung [200], for instance, define incentive contracts based on principal-agent theory to provide solutions for cooperative relationships among different partners in 4PL business. Another example is the work introduced by Bing and Zhongying [16]. They define in mathematical terms the parameters of a contract in transport & logistics collaborative business process. The solution of Augenstein *et al.* [8] introduces a platform based on service-oriented approach for managing contracts on 4PL business . The proposed solution itself is mainly focused on coordinating the business process conducted among these different partners. Nevertheless, none of the aforementioned solutions focus on the relationships among the SLAs from different scopes. The work proposed by Leitner *et al.* [105] aims at predicting SLA violations in business process. The authors consider two types of SLOs: instance-level, associated with each instance of a business process in isolation; and aggregated, representing the execution of several instances of the same type of business process. The violation prediction of aggregated SLOs is performed on values at the same SLA level, i.e., the same type of document describing the SLOs of a business process. In our case, in contrast, we need to aggregate information in the same level, but compare this information to a different level of SLA document. Goel *et al.* [74] use temporal logics of safety (DSF - Deterministic Safety Formula) to formalize the SLOs and model checking to support the monitoring conformance of SLAs. Their solution is able to detect and present to the user the occurrence of violations of the specified SLA. The aforementioned approaches do not deal with the validity between frame and specific agreements and they have limited support for expressing aggregate information in SLOs.

## 6.7 Summary

In this dissertation, a proposal to manage frame and specific agreements in non computational scenarios such as IT Development or Transport & Logistics outsourcing has been introduced using our computational techniques for managing Business Process as a Service from the previous chapters. This proposal, together iAgree specification, require extending tooling with enhanced models to support creation of specific Agreements based on frame agreements and integrating them in an analysis tool. This proposal supports the creation of specific agreements within frame agreement scope and the evaluation of their conformance and their execution. These SLA hierarchies appear naturally in other non computational scenarios so this enriched model can be the basis to solve other two-level (or even more level) SLA scenarios. We have addressed these scenarios with our proposed models from previous chapters but it could hinder the proposal of more complex hierarchies where guarantees in different layers are defined with predicates.

# Part IV

# FinalRemarks

# Conclusions and Future Work

*Yesterday is gone. Tomorrow has not yet come. We have only today. Let us begin.*

*Mother Teresa (1910-1997),*

## 7.1 Conclusions

As a main result of this dissertation, we have shown that non computational services can benefit using models and techniques from computational services, [60, 122, 153], to support their management.

The research effort was aimed by the following four research questions:

**Research question 1.** How can we model a SLA for different domains supporting its automatic management?

**Research question 2.** How can we model the relationships between frame and specific SLAs together with the relationships that appear between their terms and goals?

**Research question 3.** How can we automatically configure and adapt the underlying information systems based on the SLA for non-computational services?

**Research question 4.** How can we automate the monitoring of SLAs and their guarantees for non-computational services?

And these questions have been address with different proposals.

First, as proposal for Research Question 1, we provide an SLA model based on iAgree to describe different non-computational scenarios, including: (i) the description of service operative based on Business Process Modelling Notation (bpmn), (ii) performance metrics with a specific DSL for BPMN, PPINOT, (iii) the definition of guarantees together compensation functions based on these metrics, and (iv) configuration of processes to adapt them to different parties. Furthermore, we extend the SLA model to relate different SLAs and terms in different service scopes, enabling the description of multilayer SLAs where one SLA for a long term service can be refined for each service instance or more specific tasks and designing the operations related to (i) creation of Specific SLAs based on a Frame SLA, (ii) conformance analysis between Frame and Specific SLA. With these proposal, we address the Research Question 2.

Then, we provide a catalogue of analysis operations related to the deployment and fulfillment phases in the SLA Lifecycle. At this regarding, we design the operations related to (i) SLA monitoring and accomplishment evaluation, which addresses the Research Question 3 and (ii) service configuration and adaptation based on the SLA terms, addressing the Research Question 4.

And, lastly, in order to validate all the contributions, we provide a reference architecture as extensions of existing PAISs. Specifically, we propose: (i) A monitoring component which does not depend on the process execution system and it can be configured to existing endpoints to retrieve performance information or process a log system together the language PPINOT, to compute the performance indicators based on events, (ii) An architecture to manage the process models and handle service execution driven by their SLAs as extension of an existing BPMS, i.e. a PAIS which explicitly handles BPMN documents.

This dissertation has been applied to over 20 SLAs of non-computational services and over 300 KPIs and guarantees to validate our proposal.

## 7.2 Application scenarios

Some of the research results (total or partial) of this thesis can be applied to other domains to solve similar or different problems. Specifically:

- The modelling of real SLAs together monitoring component can be integrated as part of existing enterprise systems to improve existing ad-hoc monitoring components which have long development cycles since the SLA are signed until the monitoring component are operative (~months) and hinders the monitoring of SLAs for short term changes.
- In traditional services where analysis of the SLA accomplishments are performed after long term SLAs have terminated. In the case of transport & logistics service, this means where the number of specific agreements to be checked by a large company could reach up to 100,000 documents per month, the automated evaluation of SLA accomplishment can leverage in the analysis costs .
- In addition, the formalisation of SLAs together the analysis of their validity with the conformance operation between Frame and Specific Agreements or the evaluation of compensations functions, improve the design and correctness of SLAs which is a tedious and error-prone task. The errors in the definition of metrics

or compensations can introduce uncertain behaviours that could appear without being detected.

In summary, we can affirm that, although computational and non-computational services differ in operative, the SLA structure for them is similar. SLAs for both kind of services have to include the context where the SLA is applied, such as the involved parties in the service (usually consumer and provider, but other parties can appear), the service operative or references to related services. Also, the levels of performance guaranteed by each party, which are expressed with measurable properties, i.e. metrics, and expected goals in the form of Service Level Objectives (SLOs). Guarantees usually include compensations rules in case of underfulfilling the SLOs to enforce their accomplishment. Therefore, computational techniques can be applied to different non-computational scenarios to improve their lifecycle.

## 7.3 Limitations and extensions

Despite our proposed validation techniques have proved to be useful in many real-world SLAs, it still has some limitations that can focus our further research. Thus, we plan to extend our model so that it covers all different real-world SLAs we have found.

We next present the main limitations and possible extensions identified for the research work presented in this dissertation.

- Tool support. The contributions presented in this dissertation have been implemented into Governify ecosystem, supporting the definition and automated monitoring of SLAs. Nevertheless, other contributions of this thesis are not integrated in the ecosystem. That is the case of the adaptation of BPs based on SLAs, which is developed as an extension of a BPMS.

  *Extension*: Extend Governify with the aforementioned features.

- Configurable SLAs. In the chapter §5, we introduced the need to provide a language to configure BPs driven by SLAs and provided a simple mechanism for the current scenario. However, in current research context, there are a number of efforts related to provide different complex approaches to the configuring and adaptation of BPs.

  *Extension*: Apply configurable BPs techniques to our SLA model, so more complex adaptation scenarios can be modelled.

- Resources Management. In the chapter §5, we propose to take advantage of managing related processes instances to optimize the management of their resources. Nevertheless, we did not provide any specific technique.

  *Extension*: Apply techniques for the management of computational and human resources to optimize the performance of BPs driven by SLA.

- Abstract Process and Executable Process. In the chapter §5, we introduced the need to configure BPs driven by SLAs for different parties. In real scenarios, the processes which are executed include private details and differ from the public agreed processes. However, some scenarios, such as auditing betting products, requires that the executed processes are exactly or at least compliant with the published processes.

  *Extension*: Design and implement a compliant operation between private executable processes and public abstract processes regarding their SLA.

- SLA Hierarchy. We have proposed a materialisation of the SLA Hierarchy with a single level of SLAs and their monitoring. However, this proposal makes that the inherited terms from the Frame SLA have the form of assignation in the Specific SLA (e.g.: If the Frame SLA defines an SLO *Transit Time < 25 days*, it appears as *Transit Time = 25 days* or smaller value in the Specific SLA) but we do not support the definition and checking of a hierarchy of predicates (e.g.: *Transit Time < 25 days* in the Frame SLA and *Transit Time < 15 days* in the Specific SLA).

  *Extension*: Extend SLA Model to support a complete SLA Hierarchy which provides related constraints in SLAs at different levels.

Apart from addressing the identified limitations, there are two lines of future work. On the one hand, we want to build an SLA–aware PAIS that uses these models to improve the automation of certain tasks related to both the SLAs and their fulfillment. To this end, we plan to take advantage of the application of new computation techniques such as machine learning for the analysis of these measures so more intelligent information can be provided and the decision making to accomplish SLAs can be automated. First, these techniques can provide the identification of factors with big impact in the performance, either positive, increasing efficiency, either negative, creating bottle necks. Finding such factors can support the evaluation of the correctness of the KPI goals and compensations described in the SLA. And, second, the current performance displayed in the dashboards is analysed by business domain users to make decisions to lead to improve performance (saving resources, increasing outcomes, etc...). However, because the number of KPIs included in an SLA and the complexity of them, including the management of human resources or multiple and heterogeneous devices (as Internet of Things devices), this interpretation is too complex to be manually done. Therefore, machine learning techniques can provide performance prediction and decision making support which can no be performed with the current monitoring technique.

Regarding the model expressiveness, the model presented in this dissertation was designed to allow the definition of the analysed scenarios. As we apply the model to each new scenario, new requirements had to be considered that were not support by initial versions (exceptions like local calendars, or standard languages to define human assignment plans). However, based on the knowledge acquired in this context in our research group, we think that the definition of new scenarios can benefit from the current techniques proposed in this dissertation so we plan the application of this dissertation to new domains apart from the IT Development and Transport & Logistics.

On the other hand, we want to include additional information in SLAs to cover not only performance guarantees, but other aspects that are relevant for the customer such as compliance or audit–related issues [4]. In some scenarios, this information is covered as part of the guarantees as violation actions. However, in our dissertation we just focus on the compensation functions. At this regard, we plan the analysis of the different legal aspects and how they can be formalised in the SLA for their processing.

## 7.4 Summary

As a last summary and conclusion of this dissertation, we might affirm that we have set the basis of the automated management of non-computational services. We have extended iAgree as modelling language and Governify for the support of different proposed operations. We have left the prediction and decision taking analysis, such as resources management on runtime, out of the scope of the analysis included in this dissertation. We propose the further research to enhance this drawback and other potential enhancements of the proposals included in the Phd.

# Part V

# Appendices

# Compensables SLAs

*Never mistake activity for achievement*

*John Wooden (1910-2010),*

## A.1 Introduction

During the last years, the use of Service Level Agreements (SLA) to describe the rights and obligations of parties involved in service provisioning (typically the service consumer and the service provider) is on the rise. Amongst other information, SLAs have associated a set of service metrics on which some Service Level Objectives (SLOs) can be guaranteed to either the consumer or the provider; additionally, SLAs usually define compensations as a consequence of under or overfulfilling their SLOs (i.e. penalties or rewards). By means of these guarantees [6], one party (herein after the *guarantor*) guarantees to another party (herein after the *beneficiary*) the fulfillment of a Service Level Objective (SLO). For instance, Amazon as provider of its Elastic Compute Cloud Service (AWS EC2) has a term that guarantees its consumers an SLO such that availability $>=$ 99.95%[1]. In addition, in real-world SLAs, guarantee terms typically have associated one or more compensations that represent the consequences of underfulfilling (*penalties*) or overfulfilling (*rewards*) the SLO. For instance, Amazon is penalised with a 10% in service credits if the availability of AWS EC2 drops below 99.95%. In a previous work we coined the concept of Compensable SLAs [128] to refer to SLAs that include at least a compensation action, 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 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 Elastic Cloud service (AWS EC2) up to a limit of the 30% of the EC2 monthly billing. Such a limit helps to avoid mistakes like an automated

[1] `http://aws.amazon.com/es/ec2/sla/`

penalty without a limit that was discarded in 2005 by the UK Royal Mail company after causing a loss of 280 millions in one year and a half[2].

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 [65, 106, 112, 119, 197]. 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. As a matter of fact, this focus on automation makes it even more relevant the need to ensure that compensations are validated beforehand because they can be used to guide several aspects during the service lifecycle such as the elasticity [130], cost [71] or offerings [66].

We aim at answering the question *"How can compensations be automatically validated?"* To this end, we build on the compensable SLA model proposed in [128] to propose an automated technique to validate compensable SLAs. Furthermore, we also present a tooling support that implements our validation technique. Our proposal has been validated 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.

## A.2 Running Examples

Two of compensable SLAs found in real world scenarios are introduced as running examples. These two scenarios are a computing service and a human-driven IT support service[61].

### A.2.1 AWS EC2 SLA

Amazon Web Services (AWS) is a service catalogue that has boosted the idea 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 virtualized 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[3] that guarantees the availability of the virtual resources requested by means of the Monthly Uptime Percentage (MUP) service metric. Specifically, the SLA defines a term that guarantees the

[2]Page 3 in `http://goo.gl/o7gw6B`

[3]Available at http://aws.amazon.com/es/ec2/sla/

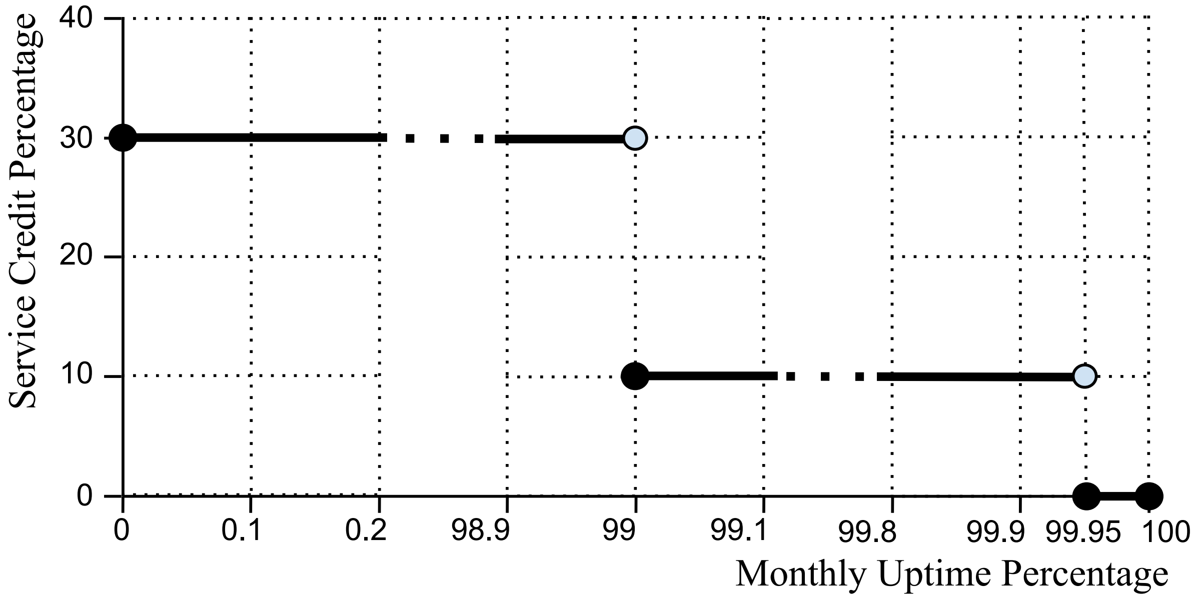


Figure A.1: Example AWS EC2: Penalties for Amazon as provider.

following SLO: *MUP* $\geq$ 99.95%. The consequences of not meeting the SLO is defined in two levels: in case the MUP drops below 99.95% and in case the MUP drops below 99% percent. Figure §A.1 depicts the penalty function [106] of this scenario that is defined as a percentage of discount in the next billing cycle (Service Credit Percentage or SCP). In this scenario Amazon is not rewarded under any circumstance. Note that in Figure §A.1 a dark point denotes the inclusion of the service metric value in the interval and a gray point means the value exclusion.

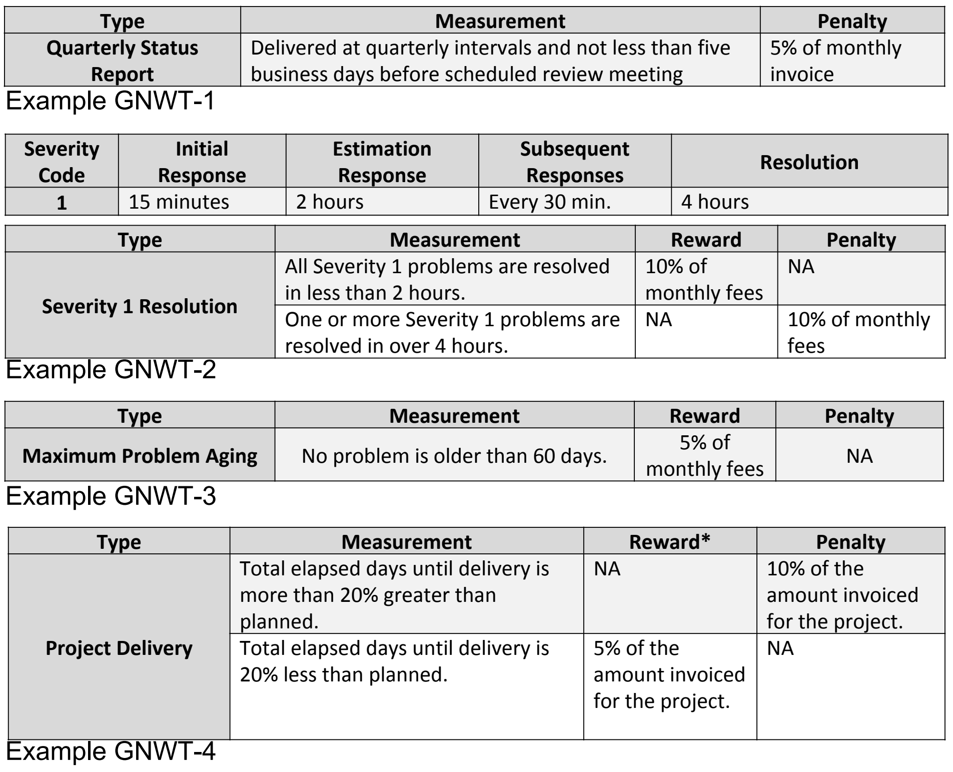
### A.2.2 GNWT SLA

The Government of the Northwest Territories (GNWT) of Canada outsources the 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[30].

Four examples of terms with at least a penalty or a reward have been extracted from its SLA template[4]. In the example GNWT-1 included in Figure §A.2 the GNWT demands the delivery of quarterly reports. The government must receive such reports not less than five days before scheduled review meetings under a penalty of 5% of monthly invoice for the IT support provider.

The example GNWT-2 of Figure §A.2 depicts specific times for different milestones that take place in the resolution of problems that have made a critical application

[4]Available at `https://goo.gl/m0duhI`



| 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

Figure A.2: Compensation actions extracted from the SLA of GNWT.

function unusable or unavailable and no workaround exists (severity 1 code). 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 than no problem is older than 60 days is included as example GNWT-3 of Figure §A.2. A term like GNWT-2 that includes both penalties and rewards is called by us as *full-compensated* term. If the term just includes either a penalty, like GNWT-1, or a reward, like GNWT-3, we call it a *half-compensated* term.

Finally, the GNWT SLA also includes a term that relates the scheduled project delivery to the real project delivery that is shown in example GNWT-4 of Figure §A.2. This term includes a penalty for the provider if the elapsed days until delivery is more than 20% greater than planned but also a reward for the provider if the elapsed days until delivery is 20% less than planned. Note that, if this latter sentence is taken literally, the reward could not make sense because it would apply if the delivery is exactly 20% less than planned and not if its less than that, e.g., 40% less. Such a problem would be solved with the automated validation proposed in Section §A.5.

## A.3 Compensation Functions

Compensation functions are defined over service metrics, or simply metrics that associate two types of compensations depending on the subject and recipient of the compensations: on the one hand, a penalty represents a compensation from the guarantor to the beneficiary and, on the other hand, a reward represents a compensation from the beneficiary to the guarantor. In this section we formalise the concept of compensation function in order to analyse some interesting properties that would support their automated validation.

### A.3.1 Core Definitions

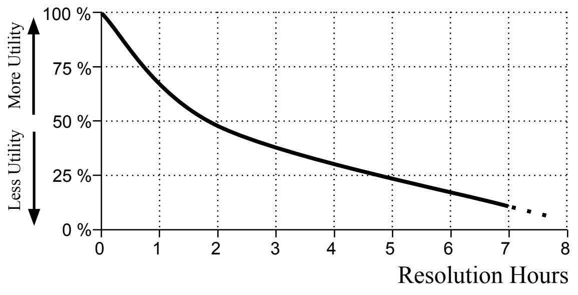
**Definition A.1 - Metric Values.**
*Let $m$ be a metric of the service, the set of all possible values of $m$ is defined as follows: $M_m = \{v_1, ..., v_n\}$.*

In the examples of Section §A.2 we find several metrics such as MUP, interventions, and urgent interventions, with the following bounded set of metric values $M_m = \{0, ..., 100\}$; and others with an infinite set of metric values such as resolution hours, or elapsed days $M_m = \{0, ..., \infty\}$.

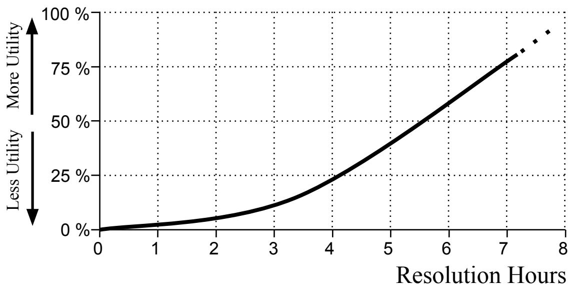
**Definition A.2 - Utility Function.**
*Let $m$ be a metric; an Utility Function for $m$ can be denoted as $U_m$, and defined as a function from $M_m$ to $\mathbb{R}$ that associates an utility to each of the values; i.e. it defines which metric values in $M_m$ are more interesting for a given party.*

For instance, Figure §A.3 includes two utility functions in §A.3(a)-(b) for the resolution hours metric of GNWT-2 example (described in Figure §A.2); an utility function for availability warranty from Amazon EC2 in §A.3(c); and an utility function for daily hours of high availability, which aims at optimizing 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 function §A.3(d)[5]. As shown in the Figure, the parties may define different kinds of utility functions such as decreasing, increasing, constant, or non-monotonic. 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.

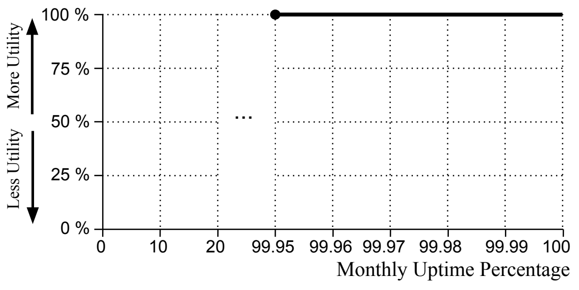
[5]Example called "horizontal demand" in [10]



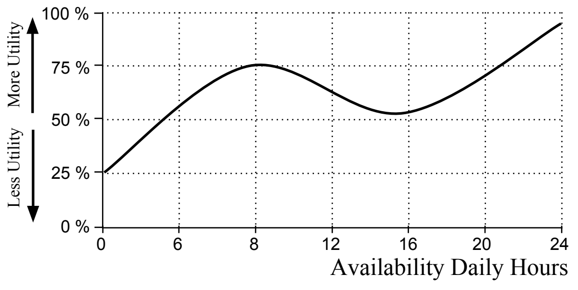
(a) Decreasing Utility Function



(b) Increasing Utility Function



(c) Constant Utility Function



(d) Non-Monotonic Utility Function

Figure A.3: Different kinds of utility functions

**Definition A.3 - Utility Precedence.**
*Let $v_1$ and $v_2$ be values of the metric values set $M_m$ of a metric $m$, and $U_m$ an utility function defined on the same metric; a precedence relation called utility precedence is defined on $M_m$ by $U_m$. Thus, we denote that $v_1$ is less interesting than $v_2$ by $v_1 \prec v_2$ when $U_m(v_1) < U_m(v_2)$.*

For instance, the example of Figure §A.3(a) may represent the utility of the beneficiary of GNWT-2. In the utility precedence defined for such utility function, the higher value of resolution hours, the less interesting it is for the beneficiary (e.g. $4 \prec 2$). On the other hand, the example of Figure §A.3(b) may represent the utility of the guarantor in the same scenario. In this case, the utility precedence defined for such a utility function establishes that a lower value of resolution hours is less interesting for the guarantor (e.g. $2 \prec 4$).

**Definition A.4 - Compensation Function.**
*Let $m$ be a metric; a Compensation Function for $m$ can be denoted as $C_m$, and defined as a function from $M_m$ to $\mathbb{R}$ that associates a compensation to each of the values. Similarly to utility functions, the compensation functions can be either decreasing, or increasing, or constant, or non-monotonic.*

As a normalised convention that is aligned with related work [106, 142] we establish a positive compensation as penalties (that should be compensated from the guarantor to the beneficiary) and negative compensations as rewards (i.e. beneficiary should compensate guarantor).

Figure §A.4 shows an example of increasing compensation function taken from the full-compensated example GNWT-2. The function denotes: (1) a reward for the guarantor 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 for the guarantor if the problems are solved in more than 4 hours. Another example is depicted in Figure §A.1. In this case, the compensation function does not take negative values, which means that only penalties are applied.

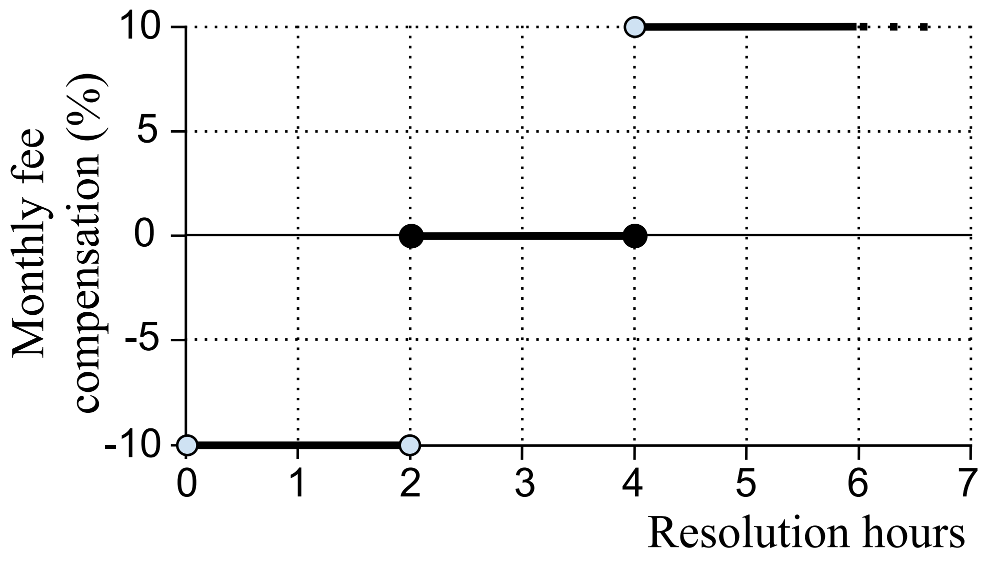


Figure A.4: Compensation function of GNWT-2 ($C_{ResolutionHours}$).

**Definition A.5 - Compensation Regions.**
*Let $C_m$ be a compensation function of a given metric m; up to three compensation regions can be defined by such compensation function, namely: penalized, rewarded, and neutral.*

$$\begin{aligned}
Penalized(C_m) &= \{v_i \in M_m \mid C_m(v_i) > 0\}\\
Neutral(C_m) &= \{v_i \in M_m \mid C_m(v_i) = 0\}\\
Rewarded(C_m) &= \{v_i \in M_m \mid C_m(v_i) < 0\}
\end{aligned}$$

Figure §A.5 shows these three potential subsets. Thus, $\forall v_i < a$ in Figure §A.5 $v_i$ is a rewarded value. $\forall v_i > b$ in Figure §A.5, $v_i$ is a penalized value. And $\forall v_i \mid a \leq v_i \leq b$ in Figure §A.5, $v_i$ is a neutral value.

## A.3.2 Validity of Compensation Functions

We consider a compensation function is valid if it fulfills two conditions.

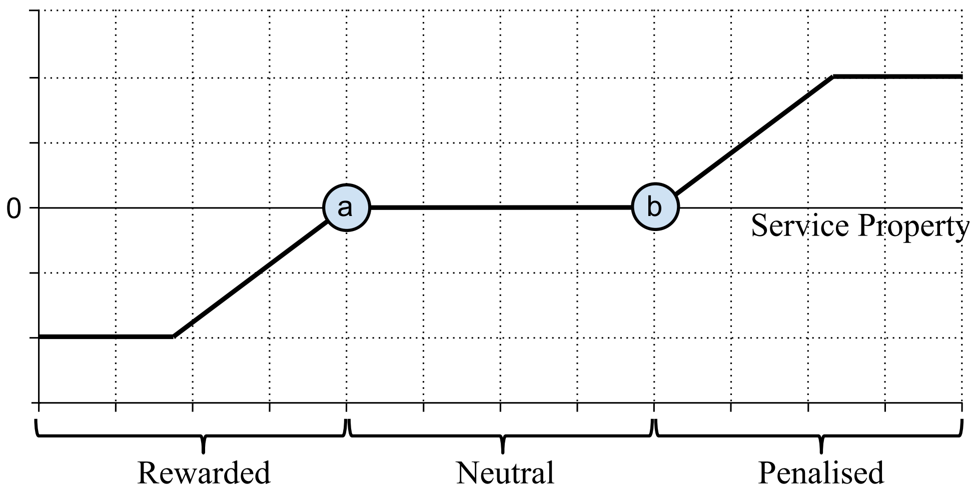


Figure A.5: A generic example of increasing compensation function

1. It is consistent with the utility function of the party to whom the guarantee is provided, i.e., the beneficiary.

2. It is saturated, i.e., it sets a limit for the maximum compensation (penalty or reward) that can be applied.

Next, we formalise these concepts using the definitions presented in the previous section.

*Property 1* (Consistent$^{U}_{CF}$). *Let m be a metric and let U be the utility function defined for m by the beneficiary, a compensation function $C_m$ for m is said to be consistent if the compensation for a less interesting value of the metric is greater or equal than the compensation for a more interesting value, according to the utility precedence defined by the utility function U.*

$$\begin{aligned} Consistent^{U}_{CF}(C_m) \Leftrightarrow \quad & \forall v_1, v_2 \in M_m \cdot v_1 \preceq v_2 \Rightarrow \\ & \Rightarrow C_m(v_1) \geq C_m(v_2) \end{aligned}$$

Let us analyse the consistency of the compensation functions of AWS EC2 and GNWT-4 examples if the beneficiary establishes in both cases that the higher the metric value, the more interesting, i.e., the utility functions $U_{MUP}$ and $U_{ElapsedDays}$ are monotonically increasing. In the AWS EC2 example, the decreasing $C_{MUP}$ depicted in Figure §A.1 is consistent with the monotonically increasing $U_{MUP}$. However, the non-monotonically increasing $C_{ElapsedDays}$ depicted in Figure §A.6 is not consistent with the $U_{ElapsedDays}$ in the GNWT-4 example because a 60% of elapsed days is a less interesting value than 80%, but its compensation is higher.

*Property 2* (Saturated). *Let m be a metric, a compensation function $C_m$ for metric m is said to be saturated with respect to a threshold $\tau = (\tau_{min}, \tau_{max})$, where $\tau_{min}, \tau_{max} \in \mathbb{R}$, if the threshold* *delimits the higher compensation, either penalty or reward.*

$$Saturated^{\tau}_{CF}(C_m) \Leftrightarrow \quad \begin{array}{l} \forall v_i \in M_m, \\ C_m(v_i) \leq \tau_{max} \wedge \\ \wedge\, C_m(v_i) \geq \tau_{min} \end{array}$$

This property prevents the definition of unbounded compensations as a boundary is defined by a threshold $\tau$. 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 finite. 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.

Based on the previous properties we can formalise the validity of a compensation function as follows:

*Property 3* ($Valid^{U,\tau}_{CF}$). *Let m be a metric, let U be an utility function defined for m by the beneficiary, and let $\tau$ be a threshold for m, a compensation function $C_m$ for metric m is said to be valid, if it is consistent according to U and saturated with respect to $\tau$.*

$$Valid^{U,\tau}_{CF}(C_m) \Leftrightarrow \quad \begin{array}{l} Consistent^{U}_{CF}(C_m) \wedge \\ \wedge\, Saturated^{\tau}_{CF}(C_m) \end{array}$$

According to this definition, the compensation functions of Figures §A.1 and §A.4 are valid. On the contrary, the compensation function of GNWT-4 example, depicted in Figure §A.6, is not consistent and therefore, not valid.

## A.4 Compensable SLA

As discussed in the introduction, a guarantee is a term of an SLA that guarantees a certain SLO to a beneficiary (e.g. *Response Time* $< 100ms$ or *Monthly Uptime Percentage* $\geq 99.95\%$). Following the terminology introduced by [128] we call a compensable guarantee to any guarantee term that also includes the concept of penalties and rewards to compensate the underfulfillment or overfulfillment of the SLO, respectively. By extension, we call compensable SLA to the type of SLA that includes at least one compensable guarantee. Next, we formalise these concepts and introduce a set of prop-

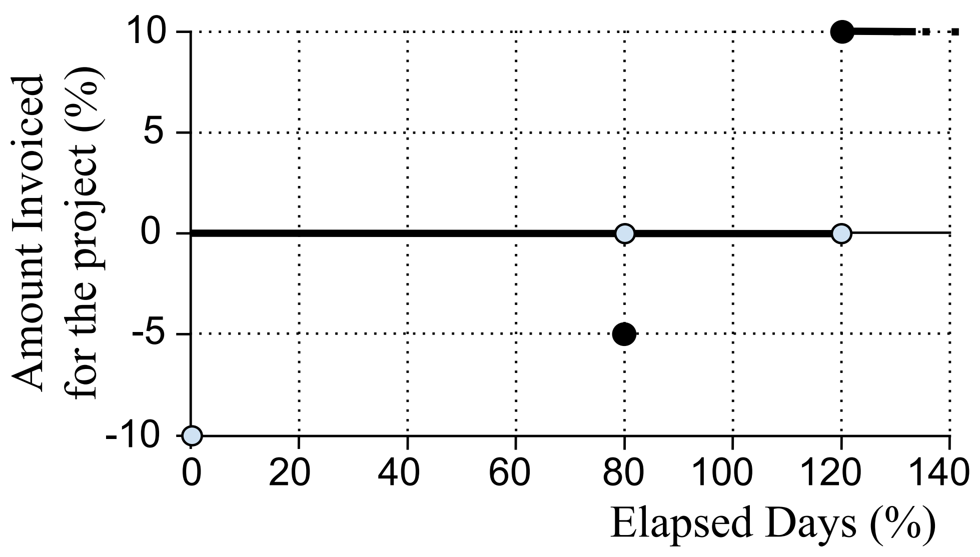


Figure A.6: Example of Inconsistent Compensation function of GNWT-4 ($C_{ElapsedDays}$).

erties to analyse the validity of compensable guarantees.

## A.4.1 Core Definitions

**Definition A.6 - Service Level Objective.**
*Let m be a service metric, $SLO_m$ is a Service Level Objective if it represents a predicate over m.*

Examples of valid SLOs include *Response Time < 100ms* or *MUP ≥ 99.95%*.

**Definition A.7 - Fulfillment Regions.**
*Let $SLO_m$ be a service level objective over the service metric m; $SLO_m$ delimits two regions over the values of m, namely fulfilled and unfulfilled.*

$$\begin{aligned} Fulfilled(SLO_m) &= \{v_i \in M_m \mid SLO_m(v_i)\} \\ Unfulfilled(SLO_m) &= \{v_i \in M_m \mid \neg SLO_m(v_i)\} \end{aligned}$$

**Definition A.8 - Compensable Guarantee.**
*Let m be a service metric, a compensable guarantee $CG_m$ defined over m is a tuple $(SLO_m, C_m)$, where $SLO_m$ is a service level objective and $C_m$ is a compensation function, that are defined over the same service metric m. With $CG_m.SLO$ we refer to the SLO of $CG_m$; and with $CG_m.C$ we refer to the compensation function of $CG_m$.*

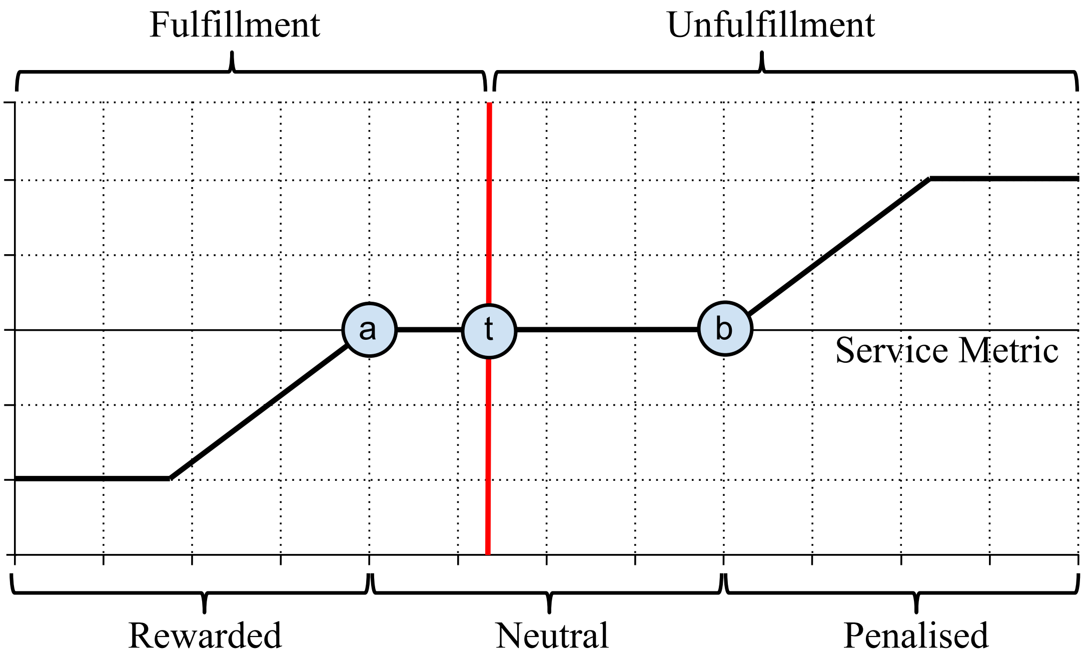


Figure A.7: A generic example of compensable guarantee showing the fulfillment and compensable regions.

---

Figure §A.7 shows a typical compensation function that depicts the relationships between the fulfillment regions delimited by the SLO and the compensation regions defined by the compensation function (c.f. Section §A.3.1). Moreover, as shown in this Figure, it is important to highlight that fulfillment 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 penalized, and similarly metric values between $a$ and $t$ are fulfilled but not rewarded.

## A.4.2 Validity of Compensable Guarantees

We consider a compensable guarantee is valid if it fulfills two conditions: its compensation function is valid according to Property 3; and its SLO and compensation function are consistent according to the following Property. *Property 4* (Consistent). *A compensable guarantee $CG_m$ is said to be consistent if the fulfillment regions are coherent with compensation regions, i.e. if there is no unfulfilled value that is rewarded, there is no fulfilled value that is penalized, and there is at least one fulfilled value that is neutral.*

$$\begin{aligned} Consistent_{CG}(CG_m) \Leftrightarrow Unfulfilled(CG_m.SLO) \\ \cap Rewarded(CG_m.C) = \varnothing \\ \wedge Fulfilled(CG_m.SLO) \cap Penalized(CG_m.C) = \varnothing \\ \wedge Fulfilled(CG_m.SLO) \cap Neutral(CG_m.C) \neq \varnothing \end{aligned}$$

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

*Property 5* ($Valid^{U,\tau}_{CG}$). *Let $m$ be a metric, let $U$ be an utility function defined for $m$ by the beneficiary, and let $\tau$ be a threshold for $m$, a compensable guarantee $CG_m$ is said to be valid if its SLO and compensation function are consistent and it contains a valid compensation function according to a threshold and the utility function of the beneficiary.*

$$\begin{aligned} Valid^{U,\tau}_{CG}(CG_m) \Leftrightarrow \quad & Consistent_{CG}(CG_m) \\ & \wedge Valid^{U,\tau}_{CF}(CG_m.C) \end{aligned}$$

## A.5 Materialising the Validity Checking

The first requirement to materialise the validity checking of compensable SLAs is to establish a specification language for the compensable SLAs themselves (cf. Section §A.5.1). Next, we discuss some extensions that facilitate the validity checking in case of missing information (cf. Section §A.5.2). Then, we propose a technique based on Constraint Satisfaction Problems (CSPs) to automate the checking (cf. Section §A.5.3).

### A.5.1 iAgree as Specification Language

The previously defined validity properties for compensable SLAs deal with several SLA elements, namely: service metrics ($M_m$), compensable guarantees ($CG$), service level objectives ($SLOs$), and compensation functions ($C$). Thus, in order to provide an automated validation technique we must use a specification language for the SLAs supporting the aforementioned SLA elements. Given such requirement, we propose a notation to specify compensations within *iAgree* [122, 126], a WS–Agreement-based [6] language with precise semantics.

Figure §A.8 depicts an excerpt of the GNWT SLA in iAgree notation 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; c.f. *InvoicePercentage* in the example). The terms are divided into *metrics* and *guarantees*. Metrics are specified by means of their schemas, which define their values ($M_m$; c.f. *elpasedDaysPercent* in the example). Therefore, schemas can be seen as a func-

```
GNWT_SLA:
...
context:
  provider: Provider
  consumer: Northwest Territories Government
  validity:
    initial: '2016-07-13T00:00:00.000Z'
  definitions:
    schemas:
      InvoicePercent: //compensation value "c"
        description: Percent affecting next monthly bill
        type: integer; unit: '%'
        minimum: -100; maximum: 100
terms:
  metrics: //service properties definitions
    elapsedDaysPercent: //metric name "m"
      schema:
        description: elapsed days until delivery (%)
        type: integer; unit: '%'
        minimum: -200; maximum: 200
  guarantees:
    id: GNWT-4
     of:
       objective: elapsedDaysPercent < 120 //SLOm
       window:
         type: static
         period: monthly
         initial: '2016-07-13T00:00:00.000Z'
       penalties:
         over: InvoicePercent:
         of:
           value: '10' //Pc[1].value
           condition: elapsedDaysPercent>=120//Pm[1].cond
       rewards:
         over: InvoicePercent:
         of:
           value: '-5' //Rc[1].value
           condition: elapsedDaysPercent==80 //Rm[1].cond
```

Figure A.8: GNWT-4 compensable guarantee term in *iAgree* syntax.

tion that returns the domain of either metrics or compensation defined as *schema(m)* = *domain(m)* and *schema(c)* = *domain(c)*, respectively. Guarantees include both: (i) the SLO defined on a service metric ($SLO_m$; c.f. *elapsedDaysPercent* $< 120$ in the example), and (ii) the Compensation Function ($C_m$). Following the WS–Agreement specification, $C_m$ 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*. 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 (we refer to it as $(P_c[i].value, P_m[i].cond)$ for penalties and $(R_c[i].value, R_m[i].cond)$ for rewards, in the following sections; e.g. ('10', *elapsedDaysPercent* $\geq 120$) in the penalty of the example). According to our definition of $C_m$, both penalty and reward are defined over the same schema. It is important to highlight that, based on the analysis of SLAs in the industry (c.f. Section §A.6), the usage of piecewise functions proves to be sufficiently expressive to model a wide range of compensations functions in real-world scenarios.

### A.5.2 Inferring the utility function and the saturation limit

The consistency checking 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; and (ii) the threshold for the saturation property. These elements are not usually public, and hence, they do not appear in the SLA. Therefore, they must be provided by service 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 so that it eliminates the need for providing the utility function in simple cases. This is useful to avoid having to define the utility function for the most simple cases.

After analysing up to 24 SLAs, 95% of their 319 compensable guarantees include simple SLOs in the form of $metric > target\_value$ or $metric < target\_value$. Therefore, the intuitive related utility can be described as a monotone linear function directly proportional (a greater value of the metric is more useful) or reverse proportional (a lower metric value is more useful) to metric values, respectively. For instance, in the SLO: $Availability \geq 99$, the utility function can be described as $U_{Availability}(x) = x$, and in the SLO $elapsedDaysPercent \leq 120$ (c.f. SLO of Figure §A.8), the utility function can be described as $U_{elapsedDaysPercent}(x) = -x$. Therefore, to avoid manual modelling, we apply this simple automatic criteria to define the utility function in our tooling. Of course, 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. `UnavailableMinutes == 0`), we transform it, with no side-effects in validation, to the SLO `UnavailableMinutes <= 0`, that allows to infer a proper utility function for the metric.

Regarding the saturation property, there are different situations where the maximum ($\tau_{max}$) and the minimum ($\tau_{min}$) values are defined in the SLA. In some cases, as in Figure §A.8, compensation is defined through constant values so the maximum and minimum possible values are in fact saturation thresholds.

In cases where there are no explicit thresholds in the SLA and they are not included in analysis, the default approach is using the domain of the compensation variable as threshold. If the compensation can reach maximum or minimum domain value, it is not saturated.

### A.5.3 Solving Technique

The SLO and compensation expressions are commonly specified by means of constraints or functions of their related service metrics. Thus, a quite straightforward way to interpret the checking of the introduced properties (c.f. sections §A.3 and §A.4) is in terms of constraints satisfaction problems (CSPs). Similar techniques have been previously applied by the authors to analyse: the conflicts between agreement terms [126],

| **Mapping for Compensation Function** $mapCF(SLO_m, P, R)$ | |
|---|---|
| Penalised Region | $(P_m[1].cond \Rightarrow P_c[1].value$ <br> $AND ... AND$ <br> $P_m[N].cond \Rightarrow P_c[N].value)$ |
| Rewarded Region | $AND$ <br> $(R_m[1].cond \Rightarrow R_c[1].value$ <br> $AND ... AND$ <br> $R_m[N].cond \Rightarrow R_c[N].value)$ |
| Neutral Region | $AND$ <br> $(NOT(P_m[1].cond\ OR ... OR\ P_m[N].cond$ <br> $OR\ R_m[1].cond\ OR ... OR\ R_m[N].cond) \Rightarrow c == 0)$ |

Table A.1: General Mapping from Compensation Function to Constraint

the SLA fulfilment at monitoring time [129], and the compliance between agreements and agreement templates [123]. CSPs are defined as a set of variables, their domains and a number of constraints, and there are a number of solvers that use satisfaction method to find a solution for the CSP, if it exists. Our approach involves mapping the iAgree SLA to a CSP and then using a CSP solver to check the validity properties. Therefore, next we thoroughly explain this mapping, which is summarized in Tables §A.1 and §A.2. In order to exemplify the CSP mapping we will use the SLA presented in Figure §A.8 (GNWT-4) as a running example in the rest of the section.

*Mapping the Compensation Function (CF):* As a common element included in every CSP used to analyse each property, we map the compensation function defined for each CG to a constraint. Table §A.1 describes this mapping. Each $(value, condition)$ pair in penalties or rewards is mapped to a constraint of the form `condition` $\Rightarrow$ `variable == value`. As neutral compensations are not usually explicit, an additional constraint, where compensation is zero outside the defined penalty and reward expressions, is added. Finally, all these constraints are combined with an `AND` operator. Note that this mapping corresponds to a well-defined piecewise function. A previous operation to check that the compensation is a mathematical well-defined function could be performed before compensation analysis.

Our running example (GNWT-4) includes a guarantee with a compensation function that expresses a penalty when metric `elapsedDaysPercent` is greater or equal than 120, and a reward when the same metric `elapsedDaysPercent` is equal to 80. Therefore, the compensation definition for GNWT4 would be mapped to the next constraint:

$$
\begin{aligned}
& mapCF(GNWT_4) = \{ \\
& \quad \{(elapsedDaysPercent >= 120 \\
& \qquad \Rightarrow InvoicePercent == 10)\ AND \\
& \quad (elapsedDaysPercent == 80 \\
& \qquad \Rightarrow InvoicePercent == -5)\ AND \\
& \quad !(elapsedDaysPercent == 80\ OR \\
& \quad\ \ elapsedDaysPercent >= 120) \\
& \qquad \Rightarrow (InvoicePercent == 0)\}
\end{aligned}
$$

This mapping is used as a base to create the CSPs to check the validity of CF and CG, which are described next. In next CSPs, the metrics and the compensation variable are the CSP variables and their corresponding domains are the CSP domains.

*CSP for Consistent Compensable Function (Property 1):* Since CSPs can only solve satisfiability problems (i.e. exists a solution) and we have to check the accomplishment of this property for all possible values, we verify this property by transforming it into an existential constraint and check that the resulting CSP is not satisfiable. Furthermore, in this CSP we duplicate the metrics variables so that we can check if two different variables with the same constraints (i.e.: same compensation function) have inconsistent compensations. For instance, if the CF is defined on metric `elapsedDaysPercent`, we define two different metrics, `elapsedDaysPercent1` and `elapsedDaysPercent2`, with the same compensation function to check the consistency. 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 CSP is satisfiable, it implies that the function is not consistent. Otherwise, it would mean that there are two values of the metric variable for which one metric value has both higher penalty and higher utility than the other value.

Considering our example, where the utility function (U) is the reverse of the previous metric, `elapsedDaysPercent` (and is inferred according previous subsection), as a greater delay in delivery is less useful for the customer (and the opposite), we get the following CSP:

$$
\begin{aligned}
& cspCCF(GNWT_4) = \{ \\
& \quad \{elapsedDaysPercent1, elapsedDaysPercent2, \\
& \qquad InvoicePercent1, InvoicePercent2\}, \\
& \quad \{int[-200..200], int[-200..200], \\
& \qquad int[-100..100], int[-100..100]\}, \\
& \quad \{mapCF(GNWT_4 - 1)\ AND\ mapCF(GNWT_4 - 2) \\
& \qquad AND\ (InvoicePercent1 > InvoicePercent2)\ AND \\
& \qquad (-elapsedDaysPercent1 > -elapsedDaysPercent2)\}\}
\end{aligned}
$$

As this problem is satisfiable, the example compensation function is *inconsistent*. This result is an effect of the wrong compensation definition, which only rewards when the value for `elapsedDaysPercent` is exactly 80, but does not reward more useful values as `elapsedDaysPercent = 60`.

*CSP for Threshold Saturated Compensable Function (Property 2):* For this property, the generation of CSP receives two additional input values, the saturation thresholds, so we only check if the metrics are outside these two values. In this case, we consider it not saturated. In our example, using 30 and -30 as input thresholds for saturation, the CSP for checking the saturation in the example compensation function is:

$$\begin{aligned}&cspTSCF(GNWT_4, 30, -30) = \{\\&\quad \{elapsedDaysPercent, InvoicePercent\},\\&\quad \{int[-200..200], int[-100..100]\},\\&\quad \{mapCF(GNWT_4)\ AND\\&\quad\ (InvoicePercent > 30\ OR\ InvoicePercent < -30)\}\}\end{aligned}$$

As the compensation value 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*.

*CSP for Valid Compensable Function (Property 3):* A compensation function is valid if it is *consistent* (Property 1) AND *saturated* (Property 2). In our example, the compensation is *invalid* (it is *saturated* but not *consistent*).

*CSP for Consistent Compensable Guarantee (Property 4):* To check the consistency of compensable guarantees, we evaluate SLO accomplishment together with the compensation function. This property involves checking one existentially quantified CSP, there exists a value that is neutral and fulfilled, and one universally quantified CSP, no value is fulfilled and penalised or unfulfilled and rewarded at the same time. Therefore, we build two different CSPs. First, if there is a neutral compensation region (NCR) for an SLO fulfilled value, and then if there is a positive compensation value (penalty) when the SLO is fulfilled or a negative compensation value (reward) when SLO is not fulfilled (consistent compensation regions, CCR). In our example, the CSPs for checking the consistency between compensation function and SLO are:

$$\begin{aligned}&cspNCR(GNWT_4) =\\&\quad \{elapsedDaysPercent, InvoicePercent\},\\&\quad \{int[-200..200], int[-100..100]\},\\&\quad \{mapCF(GNWT_4)\ AND\ (InvoicePercent == 0\\&\quad\ AND\ elapsedDaysPercent < 120)\}\end{aligned}$$

$$
\begin{aligned}
&cspCCR(GNWT_4) = mapCF(GNWT_4) \text{ AND} \\
&\quad \{elapsedDaysPercent, InvoicePercent\}, \\
&\quad \{int[-200..200], int[-100..100]\}, \\
&\quad \{mapCF(GNWT_4) \; AND \; (InvoicePercent > 0 \\
&\qquad AND \; elapsedDaysPercent < 120) \; OR \\
&\quad\; (InvoicePercent < 0 \; AND \\
&\qquad !(elapsedDaysPercent < 120)\}
\end{aligned}
$$

As cspNCR is satisfiable and cspCCR is not satisfiable for the elapsedDaysPercent metric, the compensable guarantee is *consistent*.

*CSP for Valid Compensable Function (Property 5):* If the compensable guarantee is consistent with compensation function (Property 4) and the compensation function is valid (Property 3). For our example, the compensation definition is invalid, hence the compensable guarantee is also *invalid*.

## A.6 Validation in real-world scenarios

In this section we describe how we have validated our proposal. In particular, the goal of the validation 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 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 paper, 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 evaluated up to 319 compensable guarantees that are described in natural language in 24 different scenarios[6] belonging to three different domains, namely: cloud service providers (e.g., Amazon, Google, or Rackspace), non-cloud service providers (e.g., DHL, train companies or telecommunication services), and B2B service outsourcing.

[6]Documents used to model SLAs are available at https://goo.gl/yLvxo5

### A.6.1 RQ1: Expressiveness

Regarding the expressiveness of compensable guarantees included in the 24 studied scenarios, we have found that our technique has two main limitations concerning the expressiveness of compensation functions.

Firstly, the SLO must be specified with just one service metric, then its compensations have to be based on such metric so that its related utility precedence function can be properly evaluated.

Secondly, the compensation conditions must be defined on the SLO-related metric. Additionally, constant values should be involved in compensation conditions if and only if they do not have an impact on validity conditions, expressed in section §A.4. As an example, if we have a compensation defined as `Penalty = monthlyBilling x unavailablePeriod`, the former value can be committed in validation analysis since it does not change from a billing cycle to the next.

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

### A.6.2 RQ2: Validation result

After modeling the iAgree documents of the 24 real-world scenarios and using our proposed automated validation technique we found that 9 compensable guarantees were not properly defined in the original SLAs specified in natural language.

### A.6.3 RQ3: Modeling issues

During the recent years, we have found that the use of natural language in the reviewed SLAs of real-world scenarios[7] 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:

1) Imprecise and misleading compensation conditions and/or SLO. One of our examples guarantees an availability of 99,999% 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 modeling we have considered that the 3-minutes initial exception is affecting to the aggregated failure time. Thus, we use to gather the compensation the accumulated unavailability defined as the aggregated minutes of unavailability after the first 3 divided by total minutes in the month. In addition, the SLO is not precisely defined because considering a 31-days month, the

[7]This study was comprised of 3 cases in [128], 5 in [26], and 24 in the current article. All of them are available at: `https://goo.gl/yLvxo5`.

100% of availability in minutes are 44640 minutes, and the 99.999% of availability just permits 1 minute of unavailability that won't 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% of availability) and thus, the actual SLO would be `availability` $>$ `99.991%`.

2) Usage of imprecise value. In some cases, the metric availability is expressed as a percentage with 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.

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 two aforementioned modeling problems. In the following, we provide some modeling 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 formulae for both SLO and compensations. In case of SLOs specified as equations of the form `Availability = 100` or `Unavailable Minutes = 0`, as mentioned in section §A.5.2, they should be transformed to an SLO of the following form: `Availability` $\geq$ `100` or `Unavailable Minutes` $\leq$ `0`, respectively, because this allows to infer an 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 Figure §A.8 is straightforwardly gathered from the GNWT4 term of Figure §A.2 by just considering that `elapsedDaysPercent == 100%` is the scheduled delivery date; (ii) the aforementioned imprecise compensation function definition of CloudLock scenario can be easily solved by considering the metric `monthly uptime percentage` as a kind of `availability` that is the SLO-related metric; and (iii) the exception condition of OVH scenario can be solved by considering a new metric for a kind of unavailability that does not include the first 3 minutes of unavailability, called `accumulated unavailability`. 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 imply to make more complex decisions. For instance, in GoGrid, it is guaranteed the 100% of server uptime so that the costumers are compensated with a 10,000% 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 amounted to 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.

## A.7 Related Work

As far as we know, there is no proposal to model compensation in SLAs which enables automating its validation to avoid wrong compensation definitions.

The proposal of Leitner *et al.* in [106] formalises the problem of finding the optimal set of adaptations, which minimizes the total costs arising from SLA violations and the adaptations to prevent them. In this work, a model for penalty functions is presented; this formalisation has been the starting point of the characterisation model for compensations presented in [128] and thoroughly explained in Sections §A.3 and §A.4. In [142] the mentioned authors present an approach for optimally scheduling incoming requests to virtual computing resources in the cloud, so that the sum of payments for resources and loss incurred by SLA violations is minimized. The example relates the penalty with a service property representing the duration of requests to virtual computing resources in the cloud.

Buco *et al.* propose in [119] an SLA management system, called SAM that uses penalties in a Service Level Management process to alert about potential cumulative penalty cost. Grabarnik et al. propose in [65] a model that can be used to reduce total service costs of composite services by considering a trade-off between penalty costs and fulfillment cost at a design-time choice of service suppliers. Rana *et al.* identify in [150] how SLOs may be impacted by the choice of specific penalty clauses; and they specify the penalties using WS–Agreement specification. Paschke *et al.* [144] model an SLA to automate its management. This SLA defines minimum and maximum thresholds to compensate SLAs underfulfilling or overfulfilling, but this compensation is adhoc modelled through event calls and without using any SLA specification with a precise semantics. Angelov *et al.* propose in [100] a formal representation for contracts to detect and solve different kinds of conflicts. Although the proposed contracts representation supports penalties and rewards by means of reparation clauses, they are not validated against utility functions as proposed in the current paper. Recently, some works can be found in which the authors propose models to calculate the penalty cost of cloud services [112, 197]. Specifically, Xiaoyong *et al.* propose in [197] a competitive penalty model and a corresponding penalty based profit maximization algorithm to select the appropriate cloud providers for the consumers. In turn, Maarouf *et al.* propose in [112] a different formal model considering penalties of different metrics. Although these works do not propose a model to automate the validation of penalties, both mention the need for a penalty limit, that is one of our considerations to validate compensations.

In business studies, utility function models are also analysed as they are strongly dependent on customer preferences and behaviour. Bar-Isaac *et al.* in [10] 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, Fenghui Ren *et al.* analyse in [152] how utility function is obtained from customer objective function (i.e., customers timetable preferences affect how transactions distribute through commercial opening hours).

Finally, in previous works of the authors [122, 126] the validity of SLAs is formalised by considering a set of conflict-free guarantee terms that define valid assertions that can be fulfilled. In other words, an SLA is considered as valid in those works if they are defined without conflicts between and within the SLOs (examples of conflicts are: $\textit{Response Time} < \textit{100ms AND} > \textit{100ms}$; or $\textit{MUP} < \textit{99.95\%} \Rightarrow \textit{MUP} \geq \textit{99.95\%}$). However, the notion of validity introduced in these proposals does not consider compensations.

| **Property 1: CSP for Consistent Compensable Function** | |
|---|---|
| $cspCCF(schema, SLO_m, P, R, U)$ | |
| Variables | $\{m_1, m_2, c_1, c_2\}$ |
| Domains | $\{schema_m, schema_m, schema_c, schema_c\}$ |
| Constraints | $mapCF(SLO_{m_1}, P_1, R_1)$ $AND$ $mapCF(SLO_{m_2}, P_2, R_2)$ $AND$ $(c_1 > c_2)$ $AND$ $(U(m_1) > U(m_2))$ |
| Property 1 == ! SAT ( cspCCF ) | |

| **Property 2: CSP for Threshold Saturated Compensable Function** | |
|---|---|
| $cspTSCF(schema, SLO_m, P, R, \tau_{max}, \tau_{min})$ | |
| Variables | $\{m, c\}$ |
| Domains | $\{schema_m, schema_c\}$ |
| Constraints | $mapCF(SLO_m, P, R)$ $AND$ $(c > \tau_{max})$ $OR$ $(c < \tau_{min})$ |
| Property 2 == ! SAT ( cspTSCF ) | |

| **Property 3: CSP for Valid Compensable Function (P1 & P2)** |
|---|
| Property 3 == ! SAT ( cspCCF ) AND ! SAT ( cspTSCF ) |

| **Property 4: CSP for Consistent Compensable Guarantee** | |
|---|---|
| $cspNCF(schema, SLO_m, P, R)$ | |
| Variables | $\{m, c\}$ |
| Domains | $\{schema_m, schema_c\}$ |
| Neutral | $mapCF(SLO_m, P, R)$ $AND$ $(c == 0$ $AND$ $SLO_m)$ |
| $cspCCR(schema, SLO_m, P, R)$ | |
| Variables | $\{m, c\}$ |
| Domains | $\{schema_m, schema_c\}$ |
| Constraints | $mapCF(SLO_m, P, R)$ $AND$ $((c > 0$ $AND$ $SLO_m)$ $OR$ $(c < 0$ $AND$ $NOT(SLO_m)))$ |
| Property 4 == SAT (cspNCF) AND ! SAT ( cspCCR ) | |

| **Property 5: CSP for Valid Compensable Guarantee (P3 & P4)** |
|---|
| Property 5 == ! SAT ( cspCCF ) AND ! SAT ( cspTSCF ) AND SAT ( cspNFC ) AND ! SAT ( cspCCG ) |

Table A.2: CSPs to evaluate compensation properties for Compensation Functions and Guarantee Terms

# SCU Optimization

*Alone we can do so little, together we can do so much*

*Hellen Keller (1880-1968),*

## B.1 Introduction

Organizations conduct business processes with the support of different software systems and different human teams. Teams are self-managed (individual composition, skills, task solving plan,...) to adapt tasks requirements so they can be viewed and modelled as Social Computing Units (SCUs)[56]. SCUs are virtual units representing multiple expert human-based resources that use software services for executing complex tasks, in scenarios which require adaptability, such as IT Maintenance services. In these services, heterogeneous tasks are assigned on demand to SCUs so they cannot be planned in advance. Service managers are responsible to assign SCUs considering, e.g., the current work load, tasks attributes or SCU expertise, to guarantee the best possible overall service performance. As each SCU performance impacts in overall performance, managers require each team commits certain performance objectives depending on the tasks to enforce overall service performance. These objectives are defined through the relevant performance measures and their expected values. These values can be different for each SCU.

Although different SCUs commit to similar performance measures (for tasks related to the same maintenance domain), they perform tasks with their own tools and procedures. Therefore, to measure their performance and evaluate comparably the accomplishment of each SCU commitment, managers require a system which deals with their heterogeneity. In doing so, SCU performances can be compared to make optimal decisions about the assignment of SCUs regarding to the overall and SCU-specific performance. A poor commitment accomplishment implies resources waste, delays or application of penalties. Therefore, in this paper, we propose an architecture to manage and monitor the commitments, where the processing of measures is decoupled from the extraction of performance information from *different process-aware information systems*. With this architecture, we can extend the processing mechanism to adapt different external tools independently of performance evaluation. And, lastly, the decision making regarding teams assignment can be based on this monitoring to improve the achievement of service performance commitments. Our proposal is based on the combination of different modelling languages (PPINOT [40] for defining metrics and iAgree [122] for commitments) and the adaptation of the Governify ecosystem[1] to develop a commitment management platform. Overall, we contribute developing an architecture to model, monitor and evaluate commitments from heterogeneous SCUs and describing strategies to assign SCU based on runtime performance.

The remainder of this paper is structured as follows: In the next section, the addressed service is described together with current problems. Our proposal and architecture to manage the scenario is introduced in Section §B.3 and extended in section §B.4 to describe the details of commitment monitoring and a SCU assignment strategy. In section §B.5, we describe the supporting tools developed together the validation of the assignment proposal. In section §B.6, we review the most relevant contributions to address similar problems. Finally, conclusions and possible work extensions are described in Section §B.7.

## B.2 Scenario and Research Problems

### B.2.1 Maintenance Scenario

Public organizations manage different work teams to solve issues related with the maintenance of information technologies in public services. The issues can be from different expertise fields, such as network equipment maintenance, software operation or computing infrastructure management. As teams are responsible for all stages in the resolution of their assigned issues (planning, logistics and technical solution), they are formed by multidisciplinary-skilled people. The composition of teams is self-managed to dynamically adapt to their tasks requirements so they can be seen as SCUs. This scenario is depicted in Figure §B.1.

To evaluate the service performance and support the SCU assignment decision making, the managers define templates of guarantees related to different kind of issues. Depending on their expertise fields, SCUs have to commit these guarantees (Point 1 in Figure §B.1). A guarantee includes a clear definition of the concept to measure, the related metric unit and the expected goal values. They can optionally include related penalties and rewards terms to enforce their accomplishment.

Table §B.1 describes two examples of guarantees. The former, K01, aims to avoid that a number of issues remains too long open as it affects to work loads decision, resources allocation, etc., regardless of the issue relevance. The latter, K02, focuses on the expected resolution times and depends on the issue priority. Both of them include penalties to compensate the public organization when the guarantees are not accom-

[1]http://governify.io

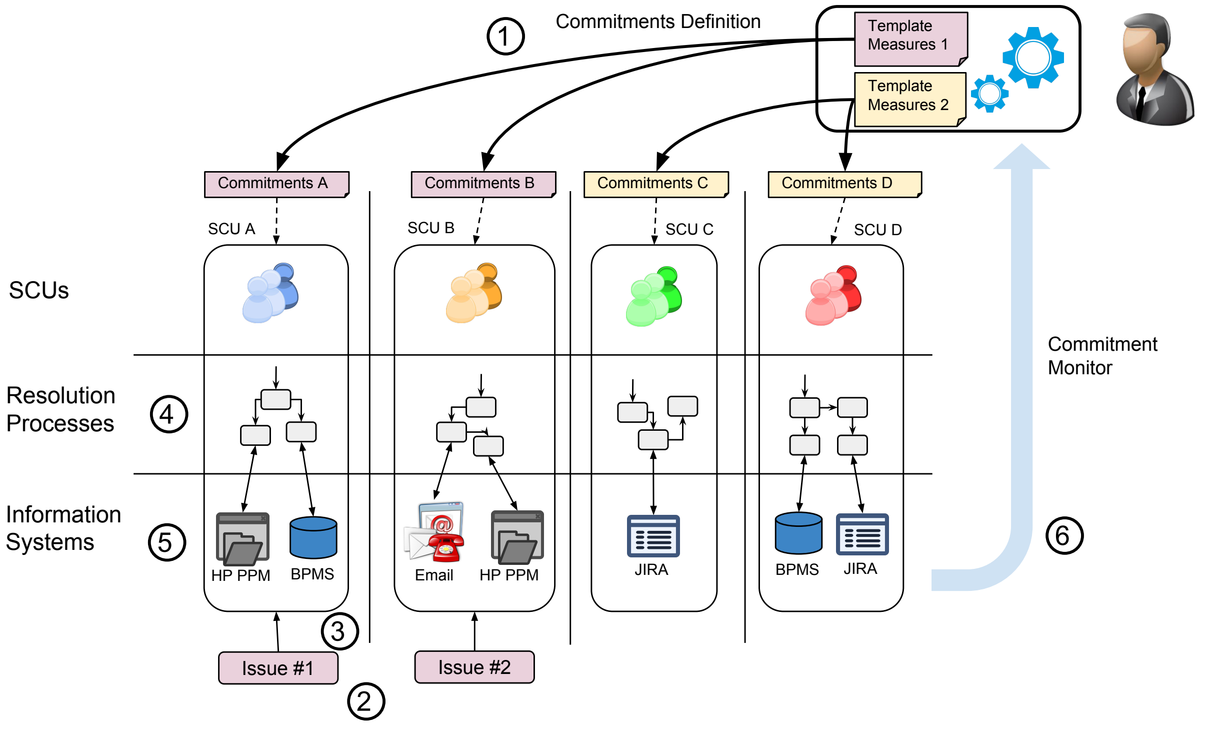


Figure B.1: Maintenance Service

plished. The penalties in these examples are expressed as percentages of the issue resolution cost but other types of penalties and rewards can be defined [162].

The service provision starts when an issue is detected and an intervention is requested (Point 2 in Figure §B.1). Prior to its resolution, each issue is assigned to a specific SCU (Point 3 in Figure §B.1). The assignment of SCUs is decided by the service managers in accordance with (i) the type of incidents, (ii) geographical location, or (iii) work load of the teams. To evaluate the commitments for each SCU, the performance of their assigned issues has to be monitored in accordance with its measures (Point 4 in Figure §B.1). Although SCUs can have similar objectives and guarantees, each one follows its own procedures to solve issues by using its own process management system, such as Jira, HP Project and Portfolio Management Software or CA Service Desk Manager and other custom software (Point 5 in Figure §B.1). Therefore, monitoring and evaluating the status of commitments requires also tackling with these heterogenous information sources (Point 6 in Figure §B.1).

## B.2.2 Research Problems

The main research goal is to provide a platform to monitor and evaluate the commitments in such above-mentioned scenarios, supporting the decision making of SCUs assignment to improve overall service performance. However, the monitoring of a

| ID | Term | Priority | Exp. Time | Objective | Penalties |
|---|---|---|---|---|---|
| K01 | Rate of Open Issues over a week | | | $\leq 2\%$ | $\leq 2{,}00\% \Rightarrow 0\%$ |
| | | | | | $\geq 2{,}00\% \Rightarrow 1\%$ |
| K02 | Rate of Closed Issues In Time | Very High | Time $< 2$h | $Rate \geq 95\%$ | $Rate \geq 95\% \Rightarrow 0\%$ |
| | | | | | $90\% \geq Rate < 95\% \Rightarrow 2\%$ |
| | | | | | $85\% \geq Rate < 90\% \Rightarrow 3\%$ |
| | | | | | $0\% \geq Rate < 85\% \Rightarrow 5\%$ |
| | | High | Time $< 4$h | $Rate \geq 90\%$ | $Rate \geq 0{,}90 \Rightarrow 0\%$ |
| | | | | | $85\% \geq Rate < 90\% \Rightarrow 1{,}5\%$ |
| | | | | | $80\% \geq Rate < 85\% \Rightarrow 2\%$ |
| | | | | | $0\% \geq Rate < 80\% \Rightarrow 3\%$ |
| | | Normal | Time $< 10$h | $Rate \geq 82{,}5\%$ | $Rate \geq 82{,}5\% \Rightarrow 0\%$ |
| | | | | | $77{,}5\% \geq Rate < 82{,}5\% \Rightarrow 0{,}75\%$ |
| | | | | | $72{,}5\% \geq Rate < 77{,}5\% \Rightarrow 1{,}25\%$ |
| | | | | | $0\% \geq Rate < 72{,}5\% \Rightarrow 1{,}75\%$ |

Table B.1: Guarantees for Long-Time Open Issues and In-Time Closed Issues

measure requires, first, handling with the custom SCU systems which are related to that measure and, then, evaluate comparably their accomplishment to present an overall picture of all SCUs performance. The commitments are reviewed periodically to update guarantee goals or update measures so a commitment model independent is required to avoid ad-hoc monitoring mechanisms and support the maintenance of the monitoring systems. To achieve this goal we need to address the following challenges:

- R1: Define and formalize commitment from different SCUs where guarantees are described independently of supporting tools and single SCU goals.
- R2: Handle with the heterogeneity of process management systems to monitor homogeneously performance measures.
- R3: Correlate the performance of different SCUs and make decisions about assignments to enforce overall commitment.

## B.3 Architecture for the management of SCUs

### B.3.1 Architecture

For the management and monitoring of commitments, we propose the architecture depicted in Figure §B.2. In this architecture, we separate the management of commitments (Commitment Manager), the computation of their related business measures

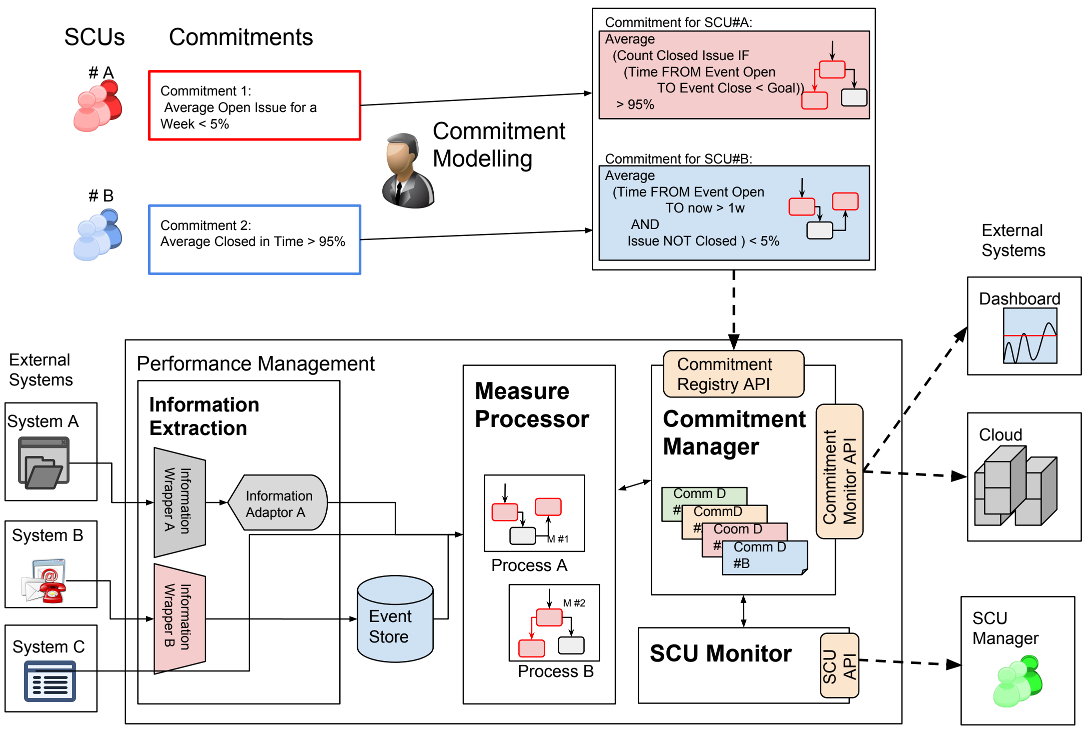


Figure B.2: Architecture for Commitments management

(Measure Processor) and the extraction of performance information (Information Extraction).

To support that commitments from different SCUs but similar guarantees (similar guarantees have identical measures but can have different goal values) are processed independently on the procedure each SCU performs, the measures are modelled with abstract process events before registering them in the system (Commitment Modelling in Figure §B.2). More details about this modelling is addressed in next section. Once they are formalized in this abstract model, commitments are registered in the system with the Commitment Manager (CM). This component also evaluates the accomplishment of the guarantees included in the commitment document (support the decision making for SCU assignment and composition). To evaluate guarantees, CM uses the Measure Processor (MP) to monitor measure values. The processing of Measures relies on the data retrieved by the Information Extraction component, which decouples the access to the performance information from the processing of such information by MP. Although MP requires the performance information on monitoring demand, there are supporting tools where information cannot be retrieved on demand, but it is provided as an event stream at runtime. In these cases, we can capture this information during resolution process in an intermediate central log or repository so MP can query from it on demand. This central store can use some standard language to describe events (such as MXML or XES[80]) or data objects.

The Commitment Manager exposes the different operations to manage and monitor

Commitments through two different APIs:

- Commitment Registry: Provides operations to create new, update existing or delete commitments in the system. Evaluation of the consistency of commitments is performed before deploying new or updated commitments.
- Commitment Monitor: The results of monitoring and evaluating KPIs can be queried through this API. This API can be consumed for different purposes: (i) Creating visual dashboard to make business management decisions, (ii) Automate claiming of penalties, etc.

Finally, the component SCU Monitor (SM) handles performance evaluation from Commitment Manager, to support decision making for SCU management, such as SCU assignment or composition. The operations computed by SM are provided through an API (SM API). In this paper, we focus on the analysis of SCU assignment which is developed in Section §B.4.

### B.3.2 Abstracting Business Measures

Although each SCU solves issues with its own mechanisms, different SCUs can be responsible to solve similar issues and commit similar guarantees, so we have to deal with their heterogeneity. In order to achieve that, the guarantees in the commitments have to be modelled with business measures independent on the issue resolution process. Therefore, first step to formalize commitments is defining these measures in terms of the abstract events related to them so we can develop the computation of measures independent of the specific SCU.

For the guarantees introduced in Section §B.2, we identify the following events:

- Rate of Closed Issues in Time: To measure this, we require the timestamp when Issues are open and when are closed to measure their duration. Therefore, we identify two relevant events: "Open Issue" and "Close Issue" and their property "Execution Time".
- Rate of Long-Time Open Issues: To measure this, we require the timestamp when Issues are open and to know IF they are closed, so we also use the previously identified events: "Open Issue" and "Close Issue" and data "Execution Time" for "Open Issue".

Any event instance or data should include the case identification data, i.e. the process instance identification, to properly process different events instances, counting processes instances, etc. The Measure Processor bases measure computation on the log of these abstract events and the wrappers map the specific events from each tool to the abstract events. This computation is further discussed in Section §B.4.

There are a number of proposals related to the abstraction of business processes, which are reviewed in the Related Work Section.

## B.4 Commitment Analytics and SCU Adaptation

In this section, we detail the model handled by the Commitment Manager component, how the measures are processed with the Measure Processor and Information Extraction components and the decision making of SCU assignment by the SCU Monitor component based on commitments accomplishment.

### B.4.1 Commitment Management

Commitment documents are composed of a set of guarantees, which are defined with their measures, target values and optional penalties. In computational domain, there are a number of proposals related to the management of commitments between service customers and providers, named service level agreements (SLAs). We use iAgree [122], a syntax consistent with WS-Agreement, a widely used standard for SLAs, which supports the definition of guarantees with penalties and rewards clauses. WS-Agreement also proposes the use of agreement templates, so the creation of new documents can be fastened by reusing and modifying existing commitment document and avoiding manual errors as we can check the consistency of commitments with their corresponding template.

To formalise measures with abstract events, we use PPINOT [40], a notation with declarative syntax to define performance indicators over processes. PPINOT enables the development of generic processors to compute measures based on the notation, as in [45, 82]. These processors are related to PPINOT data model, where the following base measures are defined: (i) the duration between two time instants (time measures); (ii) the number of times something happens (count measures); (iii) the fulfilment of certain condition in both running or finished process instances (condition measures); and (iv) the value of a certain part of a data object (data measures). The base measures can be aggregated for more than a single process case. Furthermore, new measures can be defined as a mathematical function of one or more measures.

### B.4.2 Monitoring and Processing Measures

Individual systems used by SCUs are very diverse so performance information can be stored either in accesible database or provided via some API. However, in some systems performance information persisted data is not accesible with provided API or through querying database so they have be extended to get it.

Therefore, first step in information extraction is wrapping system to have access to the performance information. Wrappers are responsible for handling with the specific mechanisms to extract information from each tool.

Once the information is retrieved, it should be consistent with the abstract events defined in guarantee measures. In some cases, it can be directly matched with their abstract events and used by Measure Processor (System B Wrapper in Figure §B.2) but, in most of the cases, there is a conceptual gap between them. In these cases, we require adapting it to our abstract events (System A Wrapper in Figure §B.2). In trivial cases, the differences between the business process and the tool workflows are just different status names (e.g.: "Open Issue" instead of "New Issue"). In other cases, one single status in business process relates to a complex workflow in external systems or, opposite, a business process fragment is just one step in the external system workflow. Some proposals related to the matching of process models are described in the Related Work section.

To evaluate measures, Measure Processor starts collecting their measure definitions in Commitment Document to process the performance information.

Besides indicators, Measure Processor requires additional information to determine how to connect to the proper information system to extract performance information. This information is not included in the commitment documents, to decouple commitment management from their measurement, so it is defined with a manifest or configuration file. Through this configuration, Measure Processor connects to the corresponding wrapper to retrieve performance information.

### B.4.3 Assigning SCUs to Issues

Currently, assignment decisions are taken considering the SCU affinity to each issue and its current availability. These two values are linked in Suitability expression and the evaluation of this expression proposes the next assigned SCU. Suitability is defined as:

$$\mathit{Suitability} = \mathit{Affinity} \times \mathit{Availability} \tag{B.1}$$

where *Affinity* is a value between 0 and 1 which relates the SCU expertise to a type of issue (for example, a network maintenance SCU has an affinity value of 1.0 to solve network issues while a microcomputer maintenance SCU can deal with these issues but its affinity value is 0.3) and other criteria regarding static suitability for a specific issue (e.g.: geographical suitability). The *Affinity* value is defined by service managers considering previous experience and it is included in the commitment.

*Availability* depends on the SCU work load. This work load can be modelled as the current issue assignments ($t$) regarding a predefined maximum work volume ($n$) per SCU:

$$Availability = 1 - Workload = 1 - \frac{t}{n} \tag{B.2}$$

where the current number of issues assigned to the SCU divided by the maximum simultaneous issues is the current work load.

Assignments just based on *Affinity* and *Availability* do not consider current SCU performance so corrective decisions cannot be automatically taken to increase the commitment achievement rate. With our proposal to automatically monitor guarantees, the automatic evaluation of their accomplishment can be used to evaluate Suitability. Therefore, we propose a new Suitability expression to assign SCUs:

$$SuitabilityG = Affinity \times Availability \times GPerformance \tag{B.3}$$

where *GPerformance* measures the success rate of Guarantees accomplishment. To define *GPerformance*, a first approach is the current average Guarantee accomplishment rate, but more complex approaches could include the number of agreed commitments (if customer has signed a high number of commitments with the same SCU reflects high "confidence") or ponder last $n$ accomplishments (e.g.: last year evaluations are more important than previous ones).

Simple *GPerformance* definition could be:

$$GPerformance = \sum_{i=0}^{n} K_i * G_i \tag{B.4}$$

where $K_i$ is a constant value to ponder the importance of Guarantee $G_i$. To normalize *GPerformance* value, $\sum_{i=0}^{n} G_i$ should be 1 and $G_i$ should be a value between 0 and 1. E.g. If the "Rate of Issues accomplished on time" has to be higher or equal than 95%, the normalized value for $G_i$ would be:

$$G_i = \begin{cases} 1 & \text{if } \textbf{Rate} \geq 95\% \\ \frac{\textbf{Rate}}{95} & \text{if } \textbf{Rate} < 95\% \end{cases}$$

## B.5 Tooling support and Validation

This work is supported with a complete component stack inside Governify Platform[2]. This stack includes the development of the proposed architecture, with Measure Processor, Commitment Manager and three Wrappers for required issue track tools. The Commitment Registry API is integrated with an editor for commitments

[2]http://www.governify.io

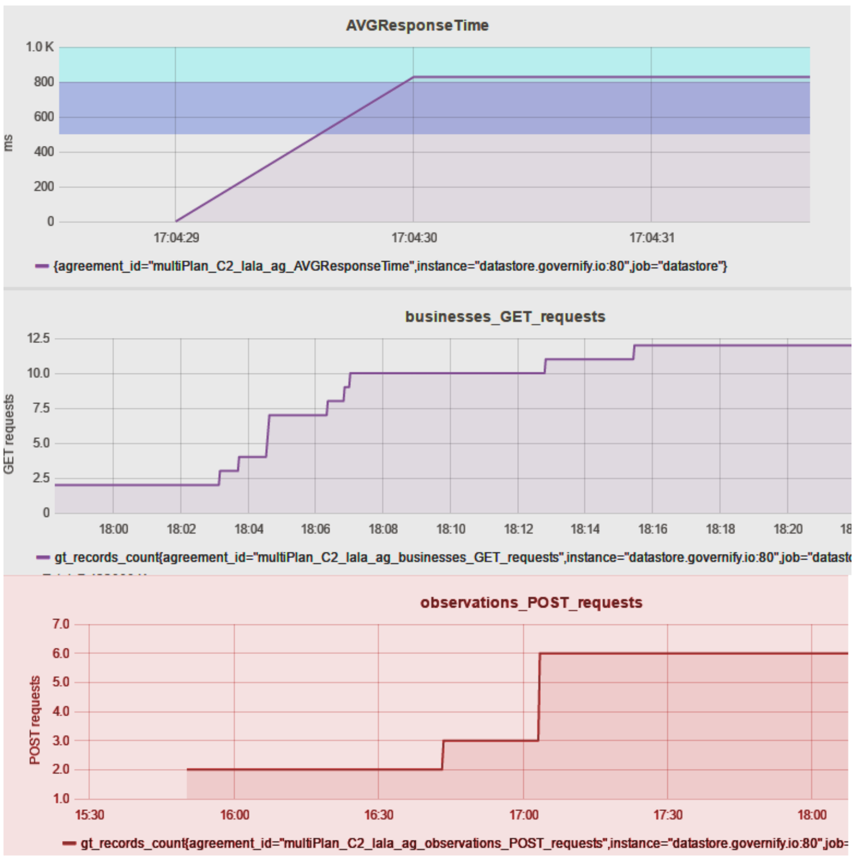


Figure B.3: Measure monitoring with Governify

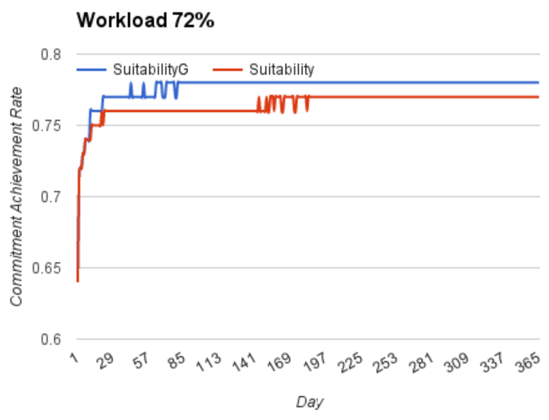
and the Commitment Monitor API is consumed by an online dashboard to graphically depict the KPIs values, the commitment status, etc.

The Governify editor supports SLA iAgree syntax and is integrated with the Commitment Registry API to create, update or delete new commitment documents in the system. In the Governify website, there is extensive information about commitment modelling and analysis with a suite of the examples used in this work. Figure §B.3 depicts the main Dashboard of Governify, where the monitoring and evaluation of commitments are graphically displayed.

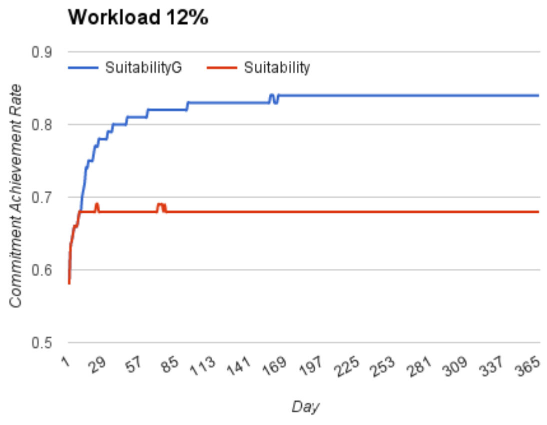
### B.5.1 Assignment Simulation

The advantage of including the performance information to increase commitment achievement is evaluated through the comparison of the commitment achievement rate with assignment based on monitoring performance or just based on availability and affinity.

We consider the following simulation for this evaluation. A set of different SCUs related to similar kind of issues (i.e. similar commitments) are assigned to appearing issues through a year period. Time to solve each Issue follows a normal distribution



(a) High Workload



(b) Low Workload

Figure B.4: Comparison of SLA Achievement rate with Similar Affinity

with different average values to goal different average work loads to reflect different situations. Average issues are defined daily so work load range for the full year significantly varies in different simulations. SCUs have different guarantee accomplishment rate (from 0.6 to 0.95) defined at the beginning of the simulation and similar maximum work loads. To simulate the impact of experience on productivity trend, we also consider that when an SCU successfully solves a number of issues, 100, it improves their accomplishment in a small percentage, 0.001.

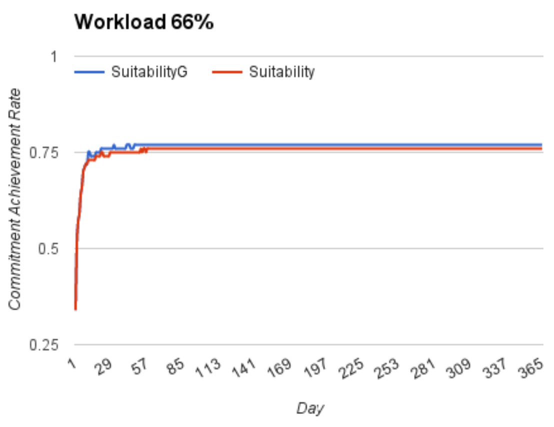
The simulation goals measuring the improvement of the commitment achievement rate considering performance with different Affinity and Work Loads. The overall achievement rate is the average of achievement rate per issue.

Figure B.4(a) (a) depicts the results of the simulation for a set of SCUs with similar affinity and a high issue load (over 70% of total maximum work load for all the SCUs). Figure B.4(b) (b) depicts simulation with similar affinity and a low issue load (about 12% of total maximum work load).

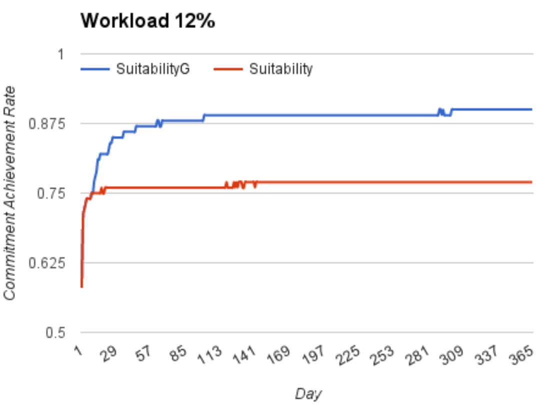
With low work load, our proposal significantly improves the commitment achievement rate as the SCU with best performance gets the most of assignments so the overall rate increases **about 12%** regarding the assignment considering just Affinity and Availability. However, when there is a high issue work load, the issues are assigned very homogeneously as differences in Guarantee accomplishment is not so significant as availability to impact in the assignment decisions. Therefore, the achievement rate is just slightly higher (between 1-2%).

We also evaluate the the case where an SCU with lower predefined Affinity has better accomplishment rate than SCUs with higher predefined Affinity (Affinity value goes from 0.7 to 1.0).

Figure §B.5 (a) depicts the results of the simulation for a set of SCUs with a high issue load (over 66% of total maximum work load for all the SCUs), and Figure §B.5



(a) High Workload



(b) Low Workload

Figure B.5: Comparison of SLA Achievement rate with Different Affinity

(b) depicts simulation with a low issue load (about 12% of total maximum workload).

With high issue work load, the commitment achievement rate is similar with or without our proposal as Availability is more significant than Guarantee Accomplishment or Affinity to assign issues. However, with low issue work load, our proposal increases significantly increases accomplishment rate regarding the assignment based on Availability, **over 15%**. Furthermore, the rate based in Availability decreases regarding the same assignment with high work load. This behaviour is due to Affinity is defined without considering current SCU performance so an SCU with bad performance but high affinity decreases commitment overall rate. This situation should be considered abnormal and it is partially solved with our proposal, as the current performance is included in the assignment decision.

## B.6 Related Work

Approaches to handle service level agreements have been studied as part of the service oriented architectures and to deal with heterogeneous processes as part of Business Process abstraction and matching. The assignment of teams to tasks based on different criteria have been also proposed in a number of works.

### B.6.1 Managing Service Level Agreements

The analysis of performance in services has been handled in several works. In [14] a quality model is proposed to describe service level agreements in computational services architectures. Developing and extending this approach, we consider iAgree, a framework to describe agreements and their terms (based on WS-Agreement recommendation) with support for the different stages in the agreement lifecycle: analysis of consistency in the agreement definition, compliance of agreement to provided templates or evaluation of service execution [129], [126]. These works are used in [45], with a notation to define process performance indicators, to propose an agreement model for business process as a service (BPaaS). In [154] and [24], Riveni and Candra describe quality models for SCUs and Hybrid Computing Units (Units composed of Software and Human). These two works consider the quality attributes of singles units to dynamically compose in SCUs. The defined attributes for these kind of units are reliability, willingness and reputation. Furthermore, this model are proposed to dynamically create human units and adapt their provision to service requirements [61]. However, they do not deal with the accomplishment of explicit commitments and how to enforce them.

### B.6.2 Business Process Abstraction and Matching

Regarding the matching of process models with other models, including sequences of events, it has been extensively studied with a number of proposals using different techniques in the last years. Van der Aalst *et al.* [178] analyse the matching between event sequence and database schemas and proposes a model to map database to event log. And, in [180], a model is proposed to classify different process cases or instances related to temporal window or properties of interest. Baier *et al.* , [9], perform the process matching based on CSP models to relate process fragments with events logs. The consistency of these fragments with the events sequences provides a metric for process matching. The work from Klinkmüller *et al.* , [96], defines and evaluates the results of applying some techniques techniques for process model matching based on labels identification. In [190], Weidlich *et al.* build a framework which a list of mechanisms ("searchers") to match two different processes. Then the results are processed to select the best "matches".In the last years, surveys in the form of contest have appeared to compare different proposals for process matching as [97]. As it mentioned before, the issue information is captured in different information systems. These systems manage information with explicit or implicit event sequence. Regarding to handling heterogenous technologies, [82] proposes different mechanisms to extract information from process-aware systems.

Smirnov *et al.* , [166], introduce a survey about proposals related to Business Process Abstraction, analysing the different aspects considered in these proposals: motivation, techniques and a list of common use cases considered for abstraction in the proposals analysed. And Eshuis *et al.* , [59], discuss about the definition of different views of a process based on privacy considerations. So they propose different process projections increasing or adding activities following different projection models: black box, grey box, glass box, open box and white box. And defines a metric to measure the relationship between the models.

### B.6.3 SCU Assignment

A number of works have proposed different strategies for team selection. Some of them, [11, 23, 55], propose a model to allocate teams based on their capabilities, work load and task requirements. Dorn *et al.* , [52], and Datta *et al.* , [36], include other aspects as the bonds or cohesiveness between individual resources to compose teams. However, none of them consider the current team performance so they can be only considered to ponder affinity and work load to assign or compose teams.

## B.7 Conclusions and Future Work

In this paper we present our architecture for the management of SCUs supported by commitments. Our architecture decouples the extraction and processing of performance information from heterogeneous tools and processes to monitor these commitments. This proposal enables the creation and monitor of new commitments avoiding the development of ad-hoc mechanisms to process their guarantees. We also detail how to support the assignment of SCU based on the relationship of guarantees in order to adapt to current performance. Our work presented in this paper is part of our ongoing research on the enforcement and governance of services in public organization. We plan to develop more complex decision taking strategies as ponder different trends or dynamically reevaluate affinity to enforce these services during runtime.

# Complete SLA Examples

## C.1 SLA from Amazon EC2 in *iAgree* syntax

Amazon provides an SLA for all its cloud services. Here we include the example of Amazon EC2, commented in the Chapter §3, which is the computation platform service. As Amazon does not provide any facility to measure the guarantee metrics, we can not include the mechanism to monitor them in the SLA.

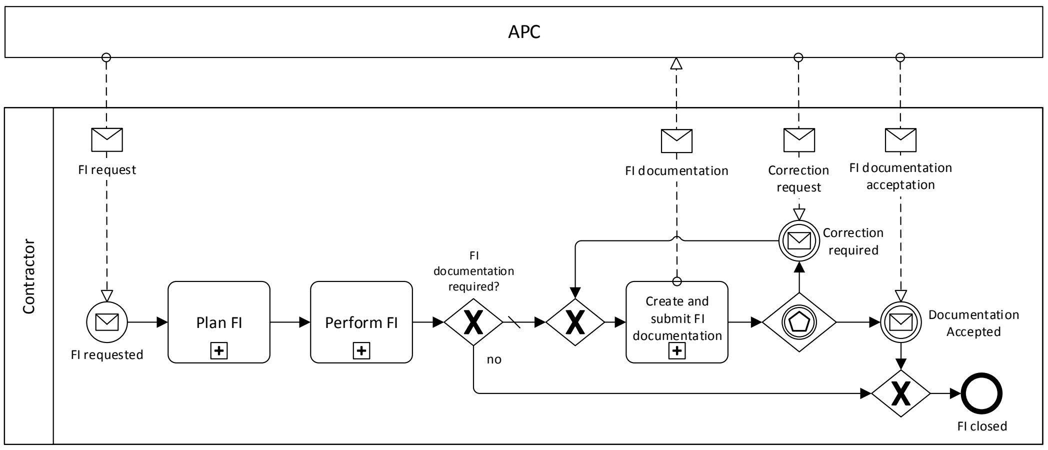
```
id: Amazon_EC2
version: '1.0'
type: agreement
 context:
  provider: Amazon
  consumer: Consumer
  validity:
    timeZone: Europe/Madrid
    initial: '2018-07-13T00:00:00.000Z'
  definitions:
    schemas:
      ServiceCredit:
        description: Percentage to decrease in the next bill
        type: integer
        unit: '%'
      scopes: { region }
terms:
  pricing:
    billing:
      period: monthly
      initial: '2017-11-12T10:35:36.000Z'
      penalties:
        - over:
            ServiceCredit:
              $ref: '#/context/definitions/schemas/...'
  metrics:
      MUP:
          schema:
            description: Monthly Uptime Percentage
            type: integer
            minimum: 0
            maximum: 100.00
            unit: '%'
            computer:
              $ref: 'http://aws/Amazon_MUP/'
  guarantees:
   - id: Amazon_GT
     scope:
        region: *
         of:
        - objective: MUP >= 99.95
          scope:
            region: *
          with:
```

```
          MUP:  {}
          window:
            type:  static
             period:  monthly
             initial:  '2016-07-13T00:00:00.000Z'
        penalties:
          - over:
               ServiceCredit:
                 $ref:  '#/context/definitions/schemas/...'
            of:
              - value:  '10'
                condition: MUP >= 99.00 && MUP < 99.95
              - value:  '30'
                condition: MUP < 99.00
```

## C.2 SLA for FI service in *iAgree* syntax

The SLA template for the Field Intervention service is exposed here. The metrics refer to computer definitions which are included below.

```
id:  FI_Service_SLA
version:  '1.0'
type:  agreement template
context:
   process:
```



```
     Calendar:  Local
   configurations:
     DocRequired:  [default:none]
     PlanFI.responsible:  [#$role$#;  default:manager]
     Timetable:  Office
     Calendar:  Local
  definitions:
    schemas:
      MonthlyFeePercentage:
        description:  Percentage  affecting  next  monthly  bill
        type:  integer
        unit:  '%'
      DocRequired:  ...
      Timetable:  ...
      Calendar:  ...
    scopes:  {}
```

```
terms:
  metrics:
    AFIP:
       schema:
         description: Average Field Interventions finished on time
      computer:
        $ref: '#/context/definitions/computers/AFIP'
        type: consumption
  guarantees:
    - id: G1
      scope:
        {region, node, priority}
      of:
        - objective: AFIP > 95%
          window:
              type: static
              period: monthly
          penalties:
              - over:
                MonthlyFeePercentage
              of:
                - value: '95 - AFIP'
                  condition: 90% <= AFIP < 95%
                - value: '10'
                  condition: AFIP < 90\%
```

The configured SLA for contractor A is exposed here.

```
id: FI_Service_SLA_ContratorA
version: '1.0'
type: agreement
context:
   ...
terms:
  configurations:
     DocRequired: high
     PlanFI.responsible: manager
  metrics:
    AFIP:
       schema:
         description: Average Field Interventions finished on time
      computer:
        $ref: '#/context/definitions/computers/AFIP'
        type: consumption
  guarantees:
    - id: G1
      scope:
        {region, node, priority}
      of:
        - objective: AFIP > 95%
          window:
              type: static
              period: monthly
          penalties:
              - over:
                MonthlyFeePercentage
              of:
                - value: '95 - AFIP'
                  condition: 90% <= AFIP < 95%
                - value: '10'
                  condition: AFIP < 90\%
```

### C.2.1 Metric examples for FI service SLA in *iAgree* syntax

Here we include an excerpt of the computer definitions for the Field Intervention SLA service. Regarding the metrics corresponding to the different levels of FI Severity, we only include the computer for Low Severity interventions, *LevelLow*, as the rest, *LevelHigh*, *LevelHigh* and *LevelCritical*, are very similar and they only differ in the precise committed times.

```
"name": "FI",
    "description": "Measures for Field Interventions SLA",
    "definitions": [
        {
            "kind": "DerivedMultiInstanceMeasure",
            "id": "AFIP",
            "name": "Percentage of Interventions finished on time",
            "description": "This metric is the number of finished interventions which
                            were accomplished on time divided by the total number of
                            finished interventions.",
            "function": " AFI_Measure / FI_Measure ",
            "includeEvidences": false,
            "usedMeasureMap": {
                "FI_Measure": {
                    "kind": "AggregatedMeasure",
                    "aggregationFunction": "Sum",
                    "baseMeasure": {
                        "kind": "CountMeasure",
                        "when": {
                            "kind": "TimeInstantCondition",
                            "appliesTo": "Event",
                            "changesToState": {
                                "EventState": {
                                    "name": "FI closed",
                                    "state": "triggered"
                                }
                            }
                        }
                    }
                },
                "AFI_Measure": {
                    "kind": "AggregatedMeasure",
                    "aggregationFunction": "Sum",
                    "baseMeasure": {
                        "kind": "DerivedSingleInstanceMeasure",
                        "function": " LevelLow AND LevelMild AND LevelHigh AND LevelCritical ",
                        "usedMeasureMap":{
                            "LevelLow": {
                                "kind": "DerivedSingleInstanceMeasure",
                                "function": " Level = Low =>     ResponseTime < 5 AND
                                                                 PresenceTime < 60 AND
                                                                 ResolutionTime < 8 AND
                                                                 DocumentationTime < 48 ",
                                "usedMeasureMap": {
                                    "Level": {
                                        "kind": "DataMeasure",
                                        "dataContentSelection": {
                                            "selection": "Severity"
                                        }
                                    },
                                    "ResponseTime": {
                                        "kind": "TimeMeasure",
                                        "unitOfMeasure": "hours",
                                        "from": {
                                            "kind": "TimeInstantCondition",
                                            "appliesTo": "Event",
                "changesToState": {
                    "EventState": {
                        "name": "FI requested",
                        "state": "triggered"
                    }
                }
            },
            "to": {
                "kind": "TimeInstantCondition",
                "appliesTo": "Event",
                "changesToState": {
                    "EventState": {
                        "name": "Plan FI",
                        "state": "active"
                    }
                }
            },
            "timeMeasureType": "LINEAR",
            "considerOnly": "${schedule}",
            "computeUnfinished": false,
            "firstTo": false
        },
        "PresenceTime": {
            "kind": "TimeMeasure",
            "unitOfMeasure": "hours",
            "from": {
                "kind": "TimeInstantCondition",
                "appliesTo": "Event",
                "changesToState": {
                    "EventState": {
                        "name": "Plan FI",
                        "state": "complete"
                    }
                }
            },
            "to": {
                "kind": "TimeInstantCondition",
                "appliesTo": "Event",
                "changesToState": {
                    "EventState": {
                        "name": "Perform FI",
                        "state": "active"
                    }
                }
            },
            "timeMeasureType": "LINEAR",
            "considerOnly": "${schedule}",
            "computeUnfinished": false,
            "firstTo": false
        },
        "ResolutionTime": {
            "kind": "TimeMeasure",
            "unitOfMeasure": "hours",
            "from": {
                "kind": "TimeInstantCondition",
                "appliesTo": "Event",
                "changesToState": {
                    "EventState": {
                        "name": "Perform FI",
                        "state": "active"
                    }
                }
            },
            "to": {
                "kind": "TimeInstantCondition",
                "appliesTo": "Event",
                "changesToState": {
```

```
                                                "EventState": {
                                                    "name": "FI closed",
                                                    "state": "triggered"
                                                }
                                            }
                                        },
                                        "timeMeasureType": "LINEAR",
                                        "singleInstanceAggFunction": "Sum",
                                        "considerOnly": "${schedule}",
                                        "computeUnfinished": false,
                                        "firstTo": false
                                    },
                                    "DocumentationTime": {
                                        "kind": "TimeMeasure",
                                        "unitOfMeasure": "hours",
                                        "from": {
                                            "kind": "TimeInstantCondition",
                                            "appliesTo": "Event",
                                            "changesToState": {
                                                "EventState": {
                                                    "name": "Create and submit FI doc",
                                                    "state": "active"
                                                }
                                            }
                                        },
                                        "to": {
                                            "kind": "TimeInstantCondition",
                                            "appliesTo": "Event",
                                            "changesToState": {
                                                "EventState": {
                                                    "name": "Create and submit FI doc",
                                                    "state": "complete"
                                                }
                                            }
                                        },
                                        "timeMeasureType": "LINEAR",
                                        "singleInstanceAggFunction": "Sum",
                                        "considerOnly": "${schedule}",
                                        "computeUnfinished": false,
                                        "firstTo": false
                                    }
                                }
                            }
                            ...
                        }
                    }
                }
            }
        }
    ]
}
```

## C.3 Frame Agreement for Work Orders service

The Frame Agreement for the IT Development outsourcing is interpreted as a SLA for BPs, where the Work Order Lifecycle is the service process, which is described as a simple Task list (as a pipeline) and for the sake of exemplification, we only include two different guarantees. One related to the aggregated metric constraint, Budget, which constraints the sum of costs of all the work orders, and an atomic metric constraint, *DevelopTime* $< 48$*hours*, which is mandatory for all the work orders.

```
id: WO_Service_SLA
version: '1.0'
type: agreement
context:
   pipeline:
  Plan
  Develop
  Document
  Deliver
terms:
  metrics:
    Budget: 10000
    DevelopTime:
       schema:
         description: Time to Develop a Work Order
      computer:
        ref: '#/context/definitions/schemas/developtime'
        type: consumption
    EstimatedCosts:
      schema:
         description: Estimated Cost for unfinished Work Orders
      computer:
        ref: '#/context/definitions/schemas/ec'
        type: consumption
    FinalCosts:
      schema:
         description: Final Cost for finished Work Orders
      computer:
        ref: '#/context/definitions/schemas/rc'
        type: consumption
  guarantees:
    - id: G1
      of:
     - objective: DevelopTime < 48
        with:
           window:
             type: static
             period: monthly
     - id: G2
      of:
     - objective: EstimatedCosts + FinalCosts < Budget
        with:
           window:
             type: static
             period: monthly
```

## C.3.1 Metric examples for Frame Agreement in *iAgree* syntax

The metrics for the Frame Agreement are listed here.

```
"name": "FA",
    "description": "Measures for Work Orders SLA",
    "definitions": [
        {
            "id": "FinalCosts",
            "kind": "AggregatedMeasure",
            "aggregationFunction": "Sum",
            "baseMeasure": {
                "kind": "DataMeasure",
                "dataContentSelection": {
                    "selection": "FinalCost"
                }
            }
        },
        {
            "id": "EstimatedCosts",
            "kind": "AggregatedMeasure",
            "aggregationFunction": "Sum",
            "baseMeasure": {
                "kind": "DataMeasure",
                "dataContentSelection": {
                    "selection": "EstimatedCost"
                }
            }
        },
        {
            "id": "DevelopTime",
            "kind": "TimeMeasure",
            "unitOfMeasure": "hours",
            "from": {
                "kind": "TimeInstantCondition",
                "appliesTo": "Event",
                "changesToState": {
                    "EventState": {
                        "name": "Develop WO",
                        "state": "active"
                    }
                }
            },
            "to": {
                "kind": "TimeInstantCondition",
                "appliesTo": "Event",
                "changesToState": {
                    "EventState": {
                        "name": "Develop WO",
                        "state": "complete"
                    }
                }
            },
            "timeMeasureType": "LINEAR",
            "considerOnly": "${schedule}",
            "computeUnfinished": false,
            "firstTo": false
        }
    ]
}
```

# Bibliography

[1] 8 European partners. Assess Grid. A project funded by the EC. https://cit-server.cit.tu-berlin.de/trac/negmgr/wiki, 2010. (page 46).

[2] W. M. P. v. d. Aalst. Verification of workflow nets. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, ICATPN '97, pages 407–426, London, UK, UK, 1997. Springer-Verlag. ISBN 3-540-63139-9. URL http://dl.acm.org/citation.cfm?id=647744.733919. (page 20).

[3] W. V. D. Aalst. The application of petri nets to workflow management, 1998. (page 20).

[4] R. Accorsi, L. Lowis, and Y. Sato. Automated Certification for Compliant Cloud-based Business Processes. *Business & Information Systems Engineering*, 3(3):145–154, Apr. 2011. ISSN 1867-0202. doi: 10.1007/s12599-011-0155-7. URL http://link.springer.com/10.1007/s12599-011-0155-7. (page 127).

[5] M. Acher, P. Collet, P. Lahire, and R. France. Managing variability in workflow with feature model composition operators. In B. Baudry and E. Wohlstadter, editors, *Software Composition*, pages 17–33, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-14046-4. (page 103).

[6] 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). Specification from the Open Grid Forum (OGF)., 03 2007. URL http://www.ogf.org/documents/GFD.107.pdf. (pages 3, 13, 31, 32, 39, 42, 78, 131, 142).

[7] C. Augenstein, A. Ludwig, and B. Franczyk. Integration of service models - preliminary results for consistent logistics service management. In *2012 Annual SRII Global Conference*, pages 100–109, July 2012. doi: 10.1109/SRII.2012.22. (pages 53, 54, 76, 77).

[8] C. Augenstein, A. Ludwig, and B. Franczyk. Integration of service models - preliminary results for consistent logistics service management. In *2012 Annual SRII Global Conference*, pages 100–109, July 2012. doi: 10.1109/SRII.2012.22. (page 118).

[9] T. Baier, A. Rogge-Solti, M. Weske, and J. Mendling. Matching of events and activities - an approach based on constraint satisfaction. In *The Practice of Enterprise Modeling*, Lecture Notes in Business Information Processing, pages 58–72. Springer Berlin Heidelberg, 12 Nov. 2014. (page 167).

[10] H. Bar-Isaac and J. Deb. What is a good reputation? career concerns with heterogeneous audiences. *International Journal of Industrial Organization*, 34: 44 – 50, 2014. ISSN 0167-7187. doi: https://doi.org/10.1016/j.ijindorg.2014.02.012. URL `http://www.sciencedirect.com/science/article/pii/S0167718714000253`. (pages 55, 76, 77, 135, 151).

[11] A. Baykasoglu, T. Dereli, and S. Das. PROJECT TEAM SELECTION USING FUZZY OPTIMIZATION APPROACH. *Cybern. Syst.*, 38(2):155–185, 2007. (page 168).

[12] J. Becker, P. Delfmann, and R. Knackstedt. *Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models*, pages 27–58. Physica-Verlag HD, Heidelberg, 2007. ISBN 978-3-7908-1966-3. doi: 10.1007/978-3-7908-1966-3_2. URL `https://doi.org/10.1007/978-3-7908-1966-3_2`. (page 95).

[13] M. Benaissa, J. Boukachour, and A. Benabdelhafid. Web service in integrated logistics information system. In *Logistics and Industrial Informatics, 2007. LINDI 2007. International Symposium on*, pages 173–178, 2007. doi: 10.1109/LINDI.2007.4343534. (page 118).

[14] P. Bianco, G. A. Lewis, and P. Merson. Service level agreements in service-oriented architecture environments. Technical report, DTIC Document, 2008. (page 166).

[15] W. Bing and L. Zhongying. Decision-making in optimizing the contract of third party logistic. In *2009 6th International Conference on Service Systems and Service Management*, pages 444–449, June 2009. doi: 10.1109/ICSSSM.2009.5174924. (page 54).

[16] W. Bing and L. Zhongying. Decision-making in optimizing the contract of third party logistic. In *Service Systems and Service Management, 2009. ICSSSM '09. 6th International Conference on*, pages 444–449, 2009. doi: 10.1109/ICSSSM.2009.5174924. (page 118).

[17] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128, May 2007. doi: 10.1109/INM.2007.374776. (page 56).

[18] C. Braga, F. Chalub, and A. Sztajnberg. A Formal Semantics for a Quality of Service Contract Language. *Electronic Notes in Theoretical Computer Science*, 203 (7):103–120, 2009. (page 31).

[19] I. Brandic, D. Music, P. Leitner, and S. Dustdar. Vieslaf framework: Enabling adaptive and versatile sla-management. In J. Altmann, R. Buyya, and O. F. Rana, editors, *Grid Economics and Business Models*, pages 60–73, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03864-8. (pages 57, 92, 93).

[20] M. G. Buscemi and U. Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In *European Symposium on Programming (ESOP), 4421 of LNCS*, pages 18–32, 2007. (page 52).

[21] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2008.12.001. URL http://www.sciencedirect.com/science/article/pii/S0167739X08001957. (page 56).

[22] R. Buyya, S. K. Garg, and R. N. Calheiros. Sla-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *2011 International Conference on Cloud and Service Computing*, pages 1–10, Dec 2011. doi: 10.1109/CSC.2011.6138522. (page 56).

[23] C. Cabanillas, M. Resinas, J. Mendling, and A. Ruiz-Cortés. Automated team selection and compliance checking in business processes. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 42–51, 2015. (page 168).

[24] M. Candra, H. L. Truong, and S. Dustdar. Analyzing reliability in hybrid compute units. In *1st IEEE International Conference on Collaboration and Internet Computing October 28 - October 30, 2015, Hangzhou, China*, 2015. (page 167).

[25] J. Cardoso, A. Barros, N. May, and U. Kylau. Towards a unified service description language for the internet of services: Requirements and first developments. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 602–609, July 2010. doi: 10.1109/SCC.2010.93. (pages 54, 76, 77).

[26] Carlos Müller et al. Supporting Compensations with WS-Agreement. In *V Congreso Espanol de Informática (CEDI 2016) - XII Jornada de Ciencia e Ingeniería de Servicios*, 2016. URL https://biblioteca.sistedes.es/wp-content/uploads/2016/09/JCIS2016_paper_9.pdf. (page 149).

[27] F. Casati, S. Castano, and M. Fugini. Managing workflow authorization constraints through active database technology. *Information Systems Frontiers*, 3(3):319–338, Sep 2001. ISSN 1572-9419. doi: 10.1023/A:1011461409620. URL https://doi.org/10.1023/A:1011461409620. (page 17).

[28] G. Chase, A. Rosenberg, R. Omar, J. Taylor, and M. Rosing. *Applying Real-World BPM in an SAP Environment*. SAP Press. Galileo Press, 2011. ISBN 9781592293438. (page 22).

[29] T. Chau, V. Muthusamy, H.-A. Jacobsen, E. Litani, A. Chan, and P. Coulthard. Automating sla modeling. In *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*, CASCON '08, pages 10:126–10:143, New York, NY, USA, 2008. ACM. doi: 10.1145/1463788.1463802. URL `http://doi.acm.org/10.1145/1463788.1463802`. (pages 53, 77, 91).

[30] M. Cho, M. Song, C. Müller, P. Fernandez, A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés. A new framework for defining realistic slas: An evidence-based approach. In *International Conference on Business Process Management*, pages 19–35. Springer, 2017. (page 133).

[31] M. Comuzzi and B. Pernici. A framework for qos-based web service contracting. *ACM Trans. Web*, 3:10:1–10:52, 2009. ISSN 1559-1131. (pages 31, 39).

[32] A. Correia, F. Brito e Abreu, and V. Amaral. Slalom: a language for sla specification and monitoring. 09 2011. (page 55).

[33] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. Snap: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 153–183, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-36180-0. (page 56).

[34] D. Daly, G. Kar, and W. H. Sanders. Modeling of service-level agreements for composed services. In M. Feridun, P. Kropf, and G. Babin, editors, *Management Technologies for E-Commerce and E-Business Applications*, pages 4–15, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-36110-7. (pages 53, 54, 76, 77).

[35] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya. A dependency-aware ontology-based approach for deploying service level agreement monitoring services in cloud. *Softw. Pract. Exper.*, 42(4):501–518, Apr. 2012. ISSN 0038-0644. doi: 10.1002/spe.1104. URL `http://dx.doi.org/10.1002/spe.1104`. (page 53).

[36] A. Datta, J. Tan Teck Yong, and A. Ventresque. T-RecS: Team recommendation system through expertise and cohesiveness. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 201–204, New York, NY, USA, 2011. ACM. (page 168).

[37] T. H. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business School Press, Boston, MA, USA, 1993. ISBN 0-87584-366-2. (page 18).

[38] D. Davide Lamanna, J. Skene, and W. Emmerich. Slang: a language for defining service level agreements. In *Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. The Ninth IEEE Workshop on Future Trends of*, pages 100–106, 2003. doi: 10.1109/FTDCS.2003.1204317. (page 50).

[39] G. Decker. Design and analysis of process choreographies. In *PhD thesis, University of Potsdam*, pages 4, 16, 19, 2009. (pages 18, 19).

[40] A. del Río-Ortega. *On the Definition and Analysis of Process Performance Indicators*. PhD thesis, University of Seville, 2012. (pages 86, 156, 161).

[41] A. del-Río-Ortega, M. Resinas, C. Cabanillas, and A. Ruiz-Cortés. On the Definition and Design-time Analysis of Process Performance Indicators. *Information Systems*, 38(4):470–490, 2012. (pages xv, 13, 23, 26, 66, 67, 78).

[42] A. del-Río-Ortega, C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. PPINOT tool suite: A performance management solution for process-oriented organisations. In *Proc. of the 11th International Conference on Service-Oriented Computing (ICSOC)*, pages 675–678, 2013. (pages 23, 67).

[43] A. del-Río-Ortega, M. Resinas, C. Cabanillas, and A. Ruiz-Cortés. Defining and analysing resource-aware process performance indicators. In *Proc. of the CAiSE'13 Forum at the 25th International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 57–64, 2013. (page 75).

[44] A. del-Río-Ortega, M. Resinas, A. Durán, and A. Ruiz-Cortés. Using templates and linguistic patterns to define process performance indicators. *Enterprise Information Systems*, In Press, 2014. doi: 10.1080/17517575.2013.867543. (pages 23, 67, 69).

[45] A. del Río-Ortega, A. M. Gutierrez, A. Durán, M. Resinas, and A. Ruiz-Cortés. Modelling service level agreements for business process outsourcing services. In *International Conference on Advance Information Systems Engineering (CAiSE)*, volume 9097 of *LNCS*, pages 485–500, 2015. (pages 161, 167).

[46] A. del-Río-Ortega, A. M. Gutiérrez, A. D. Toro, M. Resinas, and A. R. Cortés. Modelling service level agreements for business process outsourcing services. In *Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings*, pages 485–500, 2015. doi: 10.1007/978-3-319-19069-3_30. URL http://dx.doi.org/10.1007/978-3-319-19069-3_30. (page 9).

[47] A. del Río-Ortega, M. Resinas, A. Durán, B. Bernárdez, A. Ruiz-Cortés, and M. Toro. Visual ppinot: A graphical notation for process performance indicators. *Business & Information Systems Engineering*, Jun 2017. ISSN 1867-0202. doi: 10.1007/s12599-017-0483-3. URL https://doi.org/10.1007/s12599-017-0483-3. (page 65).

[48] A. del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés. Defining process performance indicators: An ontological approach. In *Proc. of the 18th International Conference on Cooperative Information Systems (CoopIS). OTM 2010, Part I*, pages 555–572, October, 2010. (page 22).

[49] A. Delgado, B. Weber, F. Ruiz, I. G. R. de Guzmán, and M. Piattini. An integrated approach based on execution measures for the continuous improvement of business processes realized by services. *Information & Software Technology*, 56 (2):134–162, 2014. (page 66).

[50] Dominic Battre et al. WS-Agreement Specification Version 1.0 Experience Document (v. gfd-e.167), 2010. OGF - Grid Resource Allocation Agreement Protocol WG. (page 46).

[51] G. T. Doran. There's a s.m.a.r.t. way to write management's goals and objectives. *Management Review*, 70(11):35–36, 1981. (page 22).

[52] C. Dorn, F. Skopik, D. Schall, and S. Dustdar. Interaction mining and skill-dependent recommendations for multi-objective team composition. *Data Knowl. Eng.*, 70(10):866–891, Oct. 2011. (page 168).

[53] H. Dresner. Business activity monitoring: BAM architecture, 2003. (page 21).

[54] M. Dumas, W. M. P. van der Aalst, and A. H. M. ter Hofstede. *Process Modeling Using Event-driven Process Chains*. Wiley, Hoboken, New Jersey, 2005. (page 26).

[55] S. Dustdar. Caramba—A Process-Aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distributed and Parallel Databases*, 15 (1):45–66. (page 168).

[56] S. Dustdar and K. Bhattacharya. The social compute unit. *Internet Computing, IEEE*, 15(3):64–69, May 2011. (page 155).

[57] J. Eder and W. Liebhart. Workflow recovery. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems*, COOPIS '96, pages 124–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7505-5. URL http://dl.acm.org/citation.cfm?id=525042.793753. (page 18).

[58] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar. Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In *2010 International Conference on High Performance Computing Simulation*, pages 48–54, June 2010. doi: 10.1109/HPCS.2010.5547150. (pages 57, 92, 93).

[59] R. Eshuis, A. Norta, O. Kopp, and E. Pitkanen. Service outsourcing with process views. *IEEE Trans. Serv. Comput.*, 8(1):136–154, Jan. 2015. (page 167).

[60] P. Fernandez. *On the Automated Procurement of Service Agreements*. PhD thesis, University of Seville, 2013. (pages 3, 39, 79, 123).

[61] P. Fernandez, H.-L. Truong, S. Dustdar, and A. Ruiz-Cortes. Programming elasticity and commitment in dynamic processes. *Programming Elasticity and Commitment in Dynamic Processes*, (2):68–74, 1 Mar. 2015. (pages 132, 167).

[62] J. Forschungszentrum, Fraunhofer, CoreGRID, and Viola. UNICORE-VIOLA, 2009. http://packcs-e0.scai.fraunhofer.de/mss-project/wsag4j/index.html. (page 46).

[63] A. Forster, G. Engels, T. Schattkowsky, and R. V. D. Straeten. Verification of business process quality constraints based on visual process patterns. In *First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE '07)*, pages 197–208, June 2007. doi: 10.1109/TASE.2007.56. (pages 76, 77, 91).

[64] Y. Foundation. *YAWL 4.1 User Manual*. 2016. (page 27).

[65] G. Grabarnik et al. Management of service process qos in a service provider - service supplier environment. In *IEEE Int. Conf. on Ent. Comp., E-Commerce, and E-Services. (CEC/EEE).*, pages 543–550, July 2007. doi: 10.1109/CEC-EEE.2007.63. (pages 132, 151).

[66] A. Gamez-Diaz, P. Fernandez, and A. Ruiz-Cortes. An analysis of restful apis offerings in the industry. In *International Conference on Service-Oriented Computing*, pages 589–604. Springer, 2017. (page 132).

[67] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu. Service level agreement based energy-efficient resource management in cloud data centers. *Computers and Electrical Engineering*, 40(5):1621 – 1633, 2014. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2013.11.001. URL http://www.sciencedirect.com/science/article/pii/S0045790613002656. (page 56).

[68] J. M. Garcia. *Improving Semantic Web Services Discovery and Ranking: A lightweight, integrated approach*. PhD thesis, University of Seville, 2012. (page 39).

[69] J. M. García, C. Pedrinaci, M. Resinas, J. Cardoso, P. Fernández, and A. Ruiz-Cortés. Linked usdl agreement: Effectively sharing semantic service level agreements on the web. In *Proceedings of the 2015 IEEE International Conference on Web Services*, ICWS '15, pages 137–144, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-7272-5. doi: 10.1109/ICWS.2015.28. URL http://dx.doi.org/10.1109/ICWS.2015.28. (page 51).

[70] J. M. García, P. Fernandez, C. Pedrinaci, M. Resinas, J. S. Cardoso, and A. R. Cortés. Modeling service level agreements with linked USDL agreement. *IEEE Trans. Services Computing*, 10(1):52–65, 2017. doi: 10.1109/TSC.2016.2593925. URL http://dx.doi.org/10.1109/TSC.2016.2593925. (pages 51, 76, 77, 91).

[71] J. M. García, O. Martín-Díaz, P. Fernandez, A. Ruiz-Cortés, and M. Toro. Automated analysis of cloud offerings for optimal service provisioning. In *International Conference on Service-Oriented Computing*, pages 331–339. Springer, 2017. (page 132).

[72] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya. Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. In *Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'11, pages 371–384, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24649-4. URL http://dl.acm.org/citation.cfm?id=2075416.2075451. (page 56).

[73] Gartner, Inc. Business Process as a Service (BPaaS). Gartner IT Glossary, 2013. URL http://www.gartner.com/it-glossary/business-process-as-a-service-bpaas. Available from http://www.gartner.com/it-glossary/business-process-as-a-service-bpaas. (page 4).

[74] N. Goel, N. Kumar, and R. K. Shyamasundar. SLA monitor: A system for dynamic monitoring of adaptive web services. In *2011 Ninth IEEE European Conference on Web Services (ECOWS)*, pages 109–116, 2011. doi: 10.1109/ECOWS.2011.22. (page 118).

[75] J. Goo. Structure of service level agreements (sla) in it outsourcing: The construct and its measurement. *Information Systems Frontiers*, 12(2):185–205, Apr 2010. ISSN 1572-9419. doi: 10.1007/s10796-008-9067-6. URL https://doi.org/10.1007/s10796-008-9067-6. (page 53).

[76] F. Gottschalk, W. M. P. Van Der Aalst, M. H. Jansen-Vullers, and M. La Rosa. Configurable workflow models. *International Journal of Cooperative Information Systems*, 17(02):177–221, 2008. doi: 10.1142/S0218843008001798. URL https://www.worldscientific.com/doi/abs/10.1142/S0218843008001798. (pages 95, 102).

[77] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.-C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321 – 343, 2004. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2003.10.007. URL http://www.sciencedirect.com/science/article/pii/S0166361503001994. Process / Workflow Mining. (pages 21, 28).

[78] P. Grubitzsch, I. Braun, H. Fichtl, T. Springer, T. Hara, and A. Schill. ML-SLA: multi-level service level agreements for highly flexible iot services. In *IEEE International Congress on Internet of Things, ICIOT 2017, Honolulu, HI, USA, June 25-30, 2017*, pages 113–120, 2017. doi: 10.1109/IEEE.ICIOT.2017.20. URL https://doi.org/10.1109/IEEE.ICIOT.2017.20. (page 54).

[79] A. Grzech and P. Rygielski. Translations of service level agreement in systems based on service oriented architecture. In R. Setchi, I. Jordanov, R. J. Howlett, and L. C. Jain, editors, *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 523–532, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15390-7. (page 53).

[80] C. W. Gunther. XES standard definition. www. xes-standard. org, 2009. *Cited on*, page 72. (page 159).

[81] A. Gutiérrez, C. Cassales Marquezan, M. Resinas, A. Metzger, A. Ruiz-Cortés, and K. Pohl. Extending ws-agreement to support automated conformity check on transport and logistics service agreements. In S. Basu, C. Pautasso, L. Zhang, and X. Fu, editors, *Service-Oriented Computing*, volume 8274 of *Lecture Notes in Computer Science*, pages 567–574. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-45004-4. doi: 10.1007/978-3-642-45005-1_47. URL http://dx.doi.org/10.1007/978-3-642-45005-1_47. (pages 4, 105, 107, 114, 118).

[82] A. M. Gutierrez, M. Resinas, A. del Río-Ortega, and A. R. Cortés. On the calculation of process performance indicators. In *XI Jornadas de Ciencia e Ingeniería de Servicios*, 09/2015 2015. (pages 161, 167).

[83] Y. h. Mai, L. x. Miao, C. p. Teo, and X. Qingqing. Geometric approach for logistics outsoursing contracting. In *2010 8th International Conference on Supply Chain Management and Information*, pages 1–7, Oct 2010. (page 54).

[84] A. Hallerbach, T. Bauer, and M. Reichert. *Configuration and Management of Process Variants*, volume 1, pages 237–255. 05 2010. (page 95).

[85] M. Hammer and J. Champy. *Reengineering the Corporation*. HarperCollins, 1999. ISBN 9780887306877. URL https://books.google.es/books?id=4UMXTn3NDisC. (pages 17, 18).

[86] A. F. M. Hani, I. V. Paputungan, and M. F. Hassan. Renegotiation in service level agreement management for a cloud-based system. *ACM Comput. Surv.*, 47 (3):51:1–51:21, Apr. 2015. ISSN 0360-0300. doi: 10.1145/2716319. URL http://doi.acm.org/10.1145/2716319. (page 56).

[87] M. Hayes and S. Shah. Hourglass: A library for incremental processing on hadoop. In *Big Data, 2013 IEEE International Conference on*, pages 742–752, Oct 2013. doi: 10.1109/BigData.2013.6691647. (page 87).

[88] M. Hedwig, S. Malkowski, and D. Neumann. Dynamic service level agreement management for efficient operation of elastic information systems. In *International Conference on Information Systems (ICIS 2011)*, 2011. (page 57).

[89] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, Mar. 2004. ISSN 0276-7783. URL http://dl.acm.org/citation.cfm?id=2017212.2017217. (page 8).

[90] Y. Hoffner, S. Field, P. Grefen, and H. Ludwig. Contract-driven creation and operation of virtual enterprises. *Computer Networks*, 37(2):111–136, 2001. (page 39).

[91] S. Huang, H. Cai, and B. Xu. A resource state-based business process control mechanism for bpm. In *2010 IEEE International Conference on Progress in Informatics and Computing*, volume 2, pages 1157–1161, Dec 2010. doi: 10.1109/PIC.2010.5687985. (page 20).

[92] S. Kailasam, N. Gnanasambandam, J. Dharanipragada, and N. Sharma. Optimizing service level agreements for autonomic cloud bursting schedulers. In *2010 39th International Conference on Parallel Processing Workshops*, pages 285–294, Sept 2010. doi: 10.1109/ICPPW.2010.54. (page 56).

[93] K. Kearney, F. Torelli, and C. Kotsokalis. Sla*: An abstract syntax for service level agreements. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 217–224, 2010. doi: 10.1109/GRID.2010.5697973. (pages 50, 76, 77, 91).

[94] A. Keller and H. Ludwig. The wsla framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11(1):57–81, 2003. ISSN 1064-7570. doi: 10.1023/A:1022445108617. URL http://dx.doi.org/10.1023/A:1022445108617. (page 38).

[95] A. Kieninger, D. Baltadzhiev, B. Schmitz, and G. Satzger. Towards Service Level Engineering for IT Services: Defining IT Services from a Line of Business Perspective. *2011 Annual SRII Global Conference*, pages 759–766, Mar. 2011. doi: 10.1109/SRII.2011.83. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5958157. (pages 54, 76, 77).

[96] C. Klinkmüller, I. Weber, J. Mendling, H. Leopold, and A. Ludwig. Increasing recall of process model matching by improved activity label matching. In *Business Process Management*, Lecture Notes in Computer Science, pages 211–218. Springer Berlin Heidelberg, 2013. (page 167).

[97] J. Kolb and Others. The process model matching contest 2015. (page 167).

[98] B. Koller, H. M. Frutos, and G. Laria. Service level agreements in brein. In P. Wieder, R. Yahyapour, and W. Ziegler, editors, *Grids and Service-Oriented Architectures for Service Level Agreements*, pages 157–165. Springer US, 2010. ISBN 978-1-4419-7320-7. URL http://dx.doi.org/10.1007/978-1-4419-7320-7_14. 10.1007/978-1-4419-7320-7_14. (page 39).

[99] C. Kotsokalis, R. Yahyapour, and M. A. Rojas Gonzalez. Modeling service level agreements with binary decision diagrams. In L. Baresi, C.-H. Chi, and J. Suzuki, editors, *Service-Oriented Computing*, pages 190–204, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10383-4. (page 53).

[100] Krasimir Angelov et al. A framework for conflict analysis of normative texts written in controlled natural language. *The Journal of Logic and Algebraic Programming*, 82(57):216 – 240, 2013. ISSN 1567-8326. doi: http://dx.doi.org/10.1016/

j.jlap.2013.03.002. URL http://www.sciencedirect.com/science/article/pii/S1567832613000143. (page 151).

[101] P. R. Krishna, K. Karlapalem, and D. Chiu. An erec framework for e-contract modeling, enactment and monitoring. *Data and Knowledge Engineering*, 51(1):31 – 58, 2004. ISSN 0169-023X. doi: 10.1016/j.datak.2004.03.006. (page 39).

[102] A. Kumar and W. Yao. Process materialization using templates and rules to design flexible process models. In G. Governatori, J. Hall, and A. Paschke, editors, *Rule Interchange and Applications*, pages 122–136, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04985-9. (page 103).

[103] L.-S. Lê, T.-V. Nguyen, T.-M. Truong, and K. Nguyen-An. Contractual specifications of business services: Modeling, formalization and proximity. In A. Hameurlain, J. Küng, R. Wagner, T. K. Dang, and N. Thoai, editors, *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXI*, pages 94–123, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg. ISBN 978-3-662-54173-9. (page 55).

[104] P. Leitner, J. Ferner, W. Hummer, and S. Dustdar. Data-driven and automated prediction of service level agreement violations in service compositions. *Distributed and Parallel Databases*, 31(3):447–470, Sep 2013. ISSN 1573-7578. doi: 10.1007/s10619-013-7125-7. URL https://doi.org/10.1007/s10619-013-7125-7. (pages 57, 92, 93).

[105] P. Leitner, J. Ferner, W. Hummer, and S. Dustdar. Data-driven and automated prediction of service level agreement violations in service compositions. *Distributed and Parallel Databases*, 31(3):447–470, Sept. 2013. ISSN 0926-8782, 1573-7578. (page 118).

[106] P. Leitner, W. Hummer, and S. Dustdar. Cost-based optimization of service compositions. *Services Computing, IEEE Transactions on*, 6(2):239–251, April 2013. ISSN 1939-1374. doi: 10.1109/TSC.2011.53. (pages 132, 133, 136, 151).

[107] L. Li and Y. Yang. *E-Business Process Modelling with Finite State Machine Based Service Agents*, pages 261–272. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-92719-8. doi: 10.1007/978-3-540-92719-8_24. URL https://doi.org/10.1007/978-3-540-92719-8_24. (page 20).

[108] X. Li and J. Du. Adaptive and attribute-based trust model for service level agreement guarantee in cloud computing. *IET Information Security*, 7(1):39–50, March 2013. ISSN 1751-8709. doi: 10.1049/iet-ifs.2012.0232. (pages 56, 92, 93).

[109] X. Li, S. J. Turner, K. H. Tong, H. M. Chan, and T. Hung. Design of an sla-driven qos management platform for provisioning multimedia personalized services. In *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, pages 1405–1409, March 2008. doi: 10.1109/WAINA.2008.256. (page 56).

[110] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592, Dec 2014. ISSN 1572-9184. doi: 10.1007/s10723-014-9314-7. URL https://doi.org/10.1007/s10723-014-9314-7. (page 56).

[111] H. Ludwig, A. Keller, A. Dan, R. King, and F. R. Web service level agreement language specification. In *Services Computing, 2008. SCC '08. IEEE International Conference on*, 2003. URL http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf. (pages 31, 41).

[112] A. Maarouf, B. E. qacimy, A. Marzouk, and A. Haqiq. A novel penalty model for managing and applying penalties in cloud computing. In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–6, Nov 2015. doi: 10.1109/AICCSA.2015.7507243. (pages 132, 151).

[113] K. Mahbub and G. Spanoudakis. Monitoring ws-agreements: An event calculus-based approach. In *Test and Analysis of Web Services, Chapter 10*, pages 265–306. 2007. (page 40).

[114] C. C. Marquezan, N. Ignaciuk, C. Alias, A. Rialland, O. Olsen, M. Turkay, A. Koestler, M. Zahlmann, S. Heyne, and M. Stollberg. D8.1 - requirements analysis and selection of technology baseline for logistics contract manager. Deliverable D8.1 of FInest Project. Available online, September 2011. URL http://www.finest-ppp.eu/files/deliverables/d08/finest_d8_1_final.pdf. (page 107).

[115] C. C. Marquezan, A. Metzger, R. Franklin, and K. Pohl. Runtime management of multi-level slas for transport and logistics services. In X. Franch, A. Ghose, G. Lewis, and S. Bhiri, editors, *Service-Oriented Computing*, volume 8831 of *Lecture Notes in Computer Science*, pages 560–574. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-45390-2. doi: 10.1007/978-3-662-45391-9_49. URL http://dx.doi.org/10.1007/978-3-662-45391-9_49. (pages 4, 118).

[116] O. Martín-Díaz. *Automated Web Services Matchmaking Using Constraint Programming (in Spanish)*. PhD thesis, University of Seville, 2007. (page 3).

[117] F. Messina, G. Pappalardo, C. Santoro, D. Rosaci, and G. M. L. Sarne. An agent based negotiation protocol for cloud service level agreements. In *2014 IEEE 23rd International WETICE Conference*, pages 161–166, June 2014. doi: 10.1109/WETICE.2014.12. (page 55).

[118] P. J. Meyer. *What would you do if you knew you could not fail? Creating S.M.A.R.T. Goals*. The Meyer Resource Group, 2003. ISBN 9780898113044. (page 22).

[119] M.J. Buco et al. Utility computing sla management based upon business objectives. *IBM Systems Journal*, 43(1):159–178, 2004. ISSN 0018-8670. doi: 10.1147/sj.431.0159. (pages 132, 151).

[120] C. Molina-Jimenez, J. Pruyne, and A. Moorsel. The role of agreements in it management software. In R. Lemos, C. Gacek, and A. Romanovsky, editors, *Architecting Dependable Systems III*, volume 3549 of *Lecture Notes in Computer Science*, pages 36–58. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28968-5. doi: 10.1007/11556169_2. URL http://dx.doi.org/10.1007/11556169_2. (page 39).

[121] B. Motik, P. F. Patel-Schneider, and B. C. Grau. OWL 2 Web Ontology Language Direct Semantics, 2009. Available from: http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/. (page 22).

[122] C. Müller. *On the Automated Analysis of WS-Agreement Documents. Applications to the Processes of Creating and Monitoring Agreements*. International dissertation, Universidad de Sevilla, 2013. (pages 3, 6, 38, 44, 47, 66, 79, 123, 142, 152, 156, 161).

[123] C. Müller, M. Resinas, and A. Ruiz-Cortés. Explaining the non-compliance between templates and agreement offers in ws-agreement*. In *Proc. of the 7th International Conference on Service Oriented Computing (ICSOC)*, volume 5900, pages 237–252, Sweden, Stockholm, Nov 2009. Springer Verlag. ISBN 3-642-10382-0. (page 145).

[124] C. Müller, J. G. Galán, A. Ruiz-Cortés, and M. Resinas. ADA: Agreement Documents Analyser. In *Proc. of the 6$^{th}$ Jornadas CientÃfico-TÃ©cnicas en Servicios Web y SOA (JSWEB 2010)*, 2010. (page 47).

[125] C. Müller, A. M. Gutiérrez, M. Resinas, P. Fernández, and A. Ruiz-Cortés. iagree studio: A platform to edit and validate WS–Agreement documents. In *Service-Oriented Computing*, Lecture Notes in Computer Science, pages 696–699. Springer Berlin Heidelberg, 2 Dec. 2013. (page 66).

[126] C. Müller, M. Resinas, and A. Ruiz-Cortés. Automated Analysis of Conflicts in WS-Agreement. *IEEE Transactions on Services Computing*, pages 1–1, Aug. 2013. ISSN 1939-1374. URL http://dx.doi.org/10.1109/TSC.2013.9. (pages 13, 32, 67, 70, 78, 142, 144, 152, 167).

[127] C. Müller, A. M. Gutiérrez, O. Martín-Díaz, M. Resinas, P. Fernandez, and A. R. Cortés. Towards a Formal Specification of SLAs with Compensations. In R. Meersman, H. Panetto, T. S. Dillon, M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea, and T. Sellis, editors, *On the Move to Meaningful Internet Systems: {OTM} 2014 Conferences - Confederated International Conferences: CoopIS, and {ODBASE} 2014, Amantea, Italy, October 27-31, 2014, Proceedings*, volume 8841 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2014. doi: 10.1007/978-3-662-45563-0_17. URL http://dx.doi.org/10.1007/978-3-662-45563-0_17. (pages 67, 71).

[128] C. Müller, A. M. Gutiérrez, O. Martín-Díaz, M. Resinas, P. Fernandez, and A. R. Cortés. Towards a formal specification of slas with compensations. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences - Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings*, pages 295–312, 2014. doi: 10.1007/978-3-662-45563-0_17. (pages 131, 132, 139, 149, 151).

[129] C. Müller, M. Oriol, X. Franch, J. Marco, M. Resinas, A. Ruiz-Cortés, and M. Rodriguez. Comprehensive Explanation of SLA Violations at Runtime. *IEEE Transactions on Services Computing*, 7(2):163–183, Sept. 2014. ISSN 1939-1374. doi: 10.1109/TSC.2013.45. (pages 70, 145, 167).

[130] C. Müller, H.-L. Truong, P. Fernandez, G. Copil, A. Ruiz-Cortés, and S. Dustdar. An elasticity-aware governance platform for cloud service delivery. In *Services Computing (SCC), 2016 IEEE International Conference on*, pages 74–81. IEEE, 2016. (page 132).

[131] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989. ISSN 0018-9219. doi: 10.1109/5.24143. (page 20).

[132] M. Netjes, H. A. Reijers, and W. M. van der Aalst. Supporting the bpm life-cycle with filenet. 2006. (page 20).

[133] T. V. Nguyen, L. S. Lê, K. Nguyen-An, and T. M. Truong. Aligning service level agreements with service-oriented enterprise architecture. In *2017 IEEE 21st International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 8–14, Oct 2017. doi: 10.1109/EDOCW.2017.11. (pages 53, 54, 77).

[134] O. Waldrich, H. Rasheed, and W. Ziegler. WS–Agreement for Java Framework (WSAG4J) by Fraunhofer SCAI Institute, and members of GRAAP-WG of the Open Grid Forum. `http://packcs-e0.scai.fraunhofer.de/wsag4j/`, 2012. (page 38).

[135] OASIS. Collaboration-Protocol Profile and Agreement Specification Version 2.0, 2002. `http://www.ebxml.org/`. (pages 31, 51, 52, 77).

[136] OASIS. ebXML Business Process Specification Schema Technical Specification v2.0.4, 2007. `http://www.ebxml.org/`. (page 51).

[137] OASIS and UN/CEFAT. Electronic business using XML (ebXML), 2007. (page 51).

[138] Object Management Group (OMG). Business process model and notation (BPMN) version 2.0, Jan 2011. Available from: http://www.omg.org/spec/BPMN/2.0/PDF. (pages 13, 78).

[139] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic ws-agreement partner selection. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 697–706, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: 10.1145/1135777.1135879. URL http://doi.acm.org/10.1145/1135777.1135879. (page 38).

[140] A. Omezzine, S. Tazi, N. Bellamine, B. Saoud, K. Drira, and G. Cooperman. Towards a dynamic multi-level negotiation framework in cloud computing. In *2015 International Conference on Cloud Technologies and Applications (CloudTech)*, pages 1–8, June 2015. doi: 10.1109/CloudTech.2015.7336999. (page 55).

[141] M. Oriol, X. Franch, J. Marco, and D. Ameller. Monitoring adaptable soa-systems using salmon. In *Workshop on Service Monitoring, Adaptation and Beyond (Mona+)*, pages 19–28, 2008. (page 40).

[142] P. Leitner et al. Cost-Efficient and Application SLA-Aware Client Side Request Scheduling in an Infrastructure-as-a-Service Cloud. In *Cloud Computing (CLOUD), IEEE 5th International Conference on*, pages 213–220, June 2012. doi: 10.1109/CLOUD.2012.21. (pages 136, 151).

[143] A. Paschke. Rbsla a declarative rule-based service level agreement language based on ruleml. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 308–314, 2005. doi: 10.1109/CIMCA.2005.1631486. (pages 52, 76, 77).

[144] A. Paschke and M. Bichler. Knowledge representation concepts for automated SLA management. 23 Nov. 2006. (page 151).

[145] C. Pedrinaci, D. Lambert, B. Wetzstein, T. van Lessen, L. Cekov, and M. Dimitrov. Sentinel: a semantic business process monitoring tool. In *international Workshop on Ontology-Supported Business Intelligence (OBI)*, pages 26–30, 2008. (page 66).

[146] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. 24:45–77, 01 2007. (page 8).

[147] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Berlin Heidelberg, 2005. ISBN 9783540243724. URL https://books.google.es/books?id=J4GqT40UsSMC. (page 95).

[148] V. Popova and A. Sharpanskykh. Formal analysis of executions of organizational scenarios based on process-oriented specifications. *Applied Intelligence*, 34:226–244, 2009. (page 22).

[149] V. Popova and A. Sharpanskykh. Modeling organizational performance indicators. *Inf. Syst.*, 35(4):505–527, 2010. (page 66).

[150] O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, and D. Cojocarasu. Managing violations in service level agreements. In *Grid Middleware and Services Chapter Title - Managing Violations in Service Level Agreements*, pages 349–358, 2008. (page 151).

[151] O. F. Rana, M. Warnier, T. B. Quillinan, and F. M. T. Brazier. Monitoring and reputation mechanisms for service level agreements. In *Grid Economics and Business Models (GECON)*, pages 125–139, 2008. (pages 36, 40).

[152] F. Ren and M. Zhang. Bilateral single-issue negotiation model considering nonlinear utility and time constraint. *Decision Support Systems*, 60(0), 2014. ISSN 0167-9236. doi: http://dx.doi.org/10.1016/j.dss.2013.05.018. URL `http://www.sciencedirect.com/science/article/pii/S0167923613001668`. (pages 55, 152).

[153] M. Resinas. *Automating the Negotiation of Agreements*. PhD thesis, University of Seville, 2008. (pages 3, 6, 40, 79, 123).

[154] M. Riveni, H. L. Truong, and S. Dustdar. On the elasticity of social compute units. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, pages 364–378, 2014. doi: 10.1007/978-3-319-07881-6_25. (page 167).

[155] M. L. Rosa, W. M. P. V. D. Aalst, M. Dumas, and F. P. Milani. Business process variability modeling: A survey. *ACM Comput. Surv.*, 50(1):2:1–2:45, Mar. 2017. ISSN 0360-0300. doi: 10.1145/3041957. URL `http://doi.acm.org/10.1145/3041957`. (page 95).

[156] A. Ruiz-Cortés. *A Semiqualitative Approach for the Automatic Management of Quality Requirements (in Spanish)*. PhD thesis, University of Seville, 2002. (page 3).

[157] A. Ruiz-Cortés, R. Corchuelo, A. Durán, and M. Toro. Automated support for quality requirements in web–services-based systems. In *Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'2001)*, pages 48–55, Bologna, Italy, Nov. 2001. IEEE CS Press. (page 51).

[158] A. Ruiz-Cortés, O. Martín-Díaz, A. Durán-Toro, and M. Toro. Improving the automatic procurement of web services using constraint programming. *International Journal of Cooperative Information Systems*, 14(4):439–468, 2005. (page 31).

[159] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual*. Addison-Wesley-Longman, 1999. ISBN 978-0-201-30998-0. (page 20).

[160] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, 2004. ISBN 0321245628. (page 20).

[161] J. Sauvé, F. Marques, A. Moura, and M. Sampaio. SLA design from a business perspective. *Ambient Networks*, pages 72–83, 2005. URL http://link.springer.com/chapter/10.1007/11568285_7. (pages 54, 76, 77).

[162] O. Scekic, H.-L. Truong, and S. Dustdar. Incentives and rewarding in social computing. *Commun. ACM*, 56(6):72–82, 1 June 2013. (page 157).

[163] A.-W. Scheer, O. Thomas, and O. Adam. *Process Modeling Using Event-driven Process Chains*, pages 119–145. Wiley, Hoboken, New Jersey, 2005. (page 19).

[164] A. Schnieders and F. Puhlmann. Variability mechanisms in e-business process families. In *9th International Conference on Business Information Systems (BIS 2006)*, pages 583–601, 01 2006. (page 95).

[165] A. Shahin and M. A. Mahbod. Prioritization of key performance indicators: An integration of analytical hierarchy process and goal setting. *International Journal of Productivity and Performance Management*, 56:226 – 240, 2007. (page 22).

[166] S. Smirnov, H. A. Reijers, M. Weske, and T. Nugteren. Business process model abstraction: a definition, catalog, and survey. *Distrib Parallel Databases*, 30(1): 63–99, 5 Jan. 2012. (page 167).

[167] S. Son, G. Jung, and S. C. Jun. An sla-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider. *The Journal of Supercomputing*, 64(2):606–637, May 2013. ISSN 1573-0484. doi: 10.1007/s11227-012-0861-z. URL https://doi.org/10.1007/s11227-012-0861-z. (page 52).

[168] V. Stantchev and C. Schröpfer. Negotiating and enforcing qos and slas in grid and cloud computing. In N. Abdennadher and D. Petcu, editors, *Advances in Grid and Pervasive Computing*, pages 25–35, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-01671-4. (page 52).

[169] W. Tan, W. Shen, L. Xu, B. Zhou, and L. Li. A business process intelligence system for enterprise process performance management. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(6):745–756, Nov 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.2001571. (page 56).

[170] Y. Tang, H. Lutfiyya, and V. Tosic. An analysis of web service sla management infrastructures based on the c-mape model. *International Journal of Business Process Integration and Management*, 4(3):209 – 218, 2009. ISSN 0004-3702. doi: 10.1504/IJBPIM.2009.030987. (page 39).

[171] S. Tata, M. Mohamed, T. Sakairi, N. Mandagere, O. Anya, and H. Ludwig. rsla: A service level agreement language for cloud services. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 415–422, June 2016. doi: 10.1109/CLOUD.2016.0062. (pages 51, 92, 93).

[172] D. B. Terry, V. Prabhakaran, R. Kotla, M. Balakrishnan, M. K. Aguilera, and H. Abu-Libdeh. Consistency-based service level agreements for cloud storage. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 309–324, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2388-8. doi: 10.1145/2517349.2522731. URL http://doi.acm.org/10.1145/2517349.2522731. (page 55).

[173] Y. Tokairin, K. Yamanaka, H. Takahashi, T. Suganuma, and N. Shiratori. An effective qos control scheme for ubiquitous services based on context information management. In *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, pages 619–625, July 2007. doi: 10.1109/CEC-EEE.2007.20. (page 56).

[174] U. University. Decentralized, cross-middleware Grid Job Submission Service (JSS), 2010. (page 46).

[175] R. B. Uriarte, F. Tiezzi, and R. D. Nicola. Slac: A formal service-level-agreement language for cloud computing. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 419–426, Dec 2014. doi: 10.1109/UCC.2014.53. (page 53).

[176] W. van den Heuvel. Survey on business process management. Technical report, 2008. (pages 19, 20).

[177] W. Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer Science & Business Media, 2011. (page 28).

[178] W. M. P. van der Aalst. Extracting event data from databases to unleash process mining. (page 167).

[179] W. M. P. van der Aalst. Business process configuration in the cloud: How to support and analyze multi-tenant processes? In *9th IEEE European Conference on Web Services, ECOWS 2011, Lugano, Switzerland, September 14-16, 2011*, pages 3–10, 2011. doi: 10.1109/ECOWS.2011.8. (pages 4, 61, 79).

[180] W. M. P. van der Aalst. Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *Asia Pacific Business Process Management*, Lecture Notes in Business Information Processing, pages 1–22. Springer International Publishing, 2013. (page 167).

[181] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business process management: A survey. In *Business Process Management*, pages 1–12, 2003. (pages 18, 20).

[182] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches.

*Data Knowl. Eng.*, 47(2):237–267, Nov. 2003. ISSN 0169-023X. doi: 10.1016/S0169-023X(03)00066-1. URL http://dx.doi.org/10.1016/S0169-023X(03)00066-1. (pages 21, 28).

[183] W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, and M. H. Jansen-Vullers. Configurable process models as a basis for reference modeling. In C. J. Bussler and A. Haller, editors, *Business Process Management Workshops*, pages 512–518, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-32596-3. (pages 95, 102).

[184] W. M. P. van der Aalst, M. Pesic, and M. Song. Beyond process mining: From the past to present and future. In *Proc. of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 38–52, 2010. (pages 21, 28).

[185] C. Vercellis. Business intelligence: Data mining and optimization for decision making. 03 2009. (page 21).

[186] J. Vonk and P. Grefen. Cross-organizational transaction support for e-services in virtual enterprises. *Distributed and Parallel Databases*, 14:137–172, 2003. ISSN 0926-8782. URL http://dx.doi.org/10.1023/A:1024884626434. 10.1023/A:1024884626434. (page 39).

[187] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In M. G. Jaatun, G. Zhao, and C. Rong, editors, *Cloud Computing*, pages 254–265, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10665-1. (page 55).

[188] W3C. Xml schema part 0: Primer second edition. http://www.w3.org/TR/xmlschema-0/. (page 40).

[189] B. Weber, S. Sadiq, and M. Reichert. Beyond rigidity – dynamic process lifecycle support. *Computer Science - Research and Development*, 23(2):47–65, May 2009. ISSN 0949-2925. doi: 10.1007/s00450-009-0069-5. URL https://doi.org/10.1007/s00450-009-0069-5. (page 26).

[190] M. Weidlich, R. Dijkman, and J. Mendling. The ICoP framework: Identification of correspondences between process models. In *Advanced Information Systems Engineering*, Lecture Notes in Computer Science, pages 483–498. Springer Berlin Heidelberg, 7 June 2010. (page 167).

[191] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007. ISBN 978-3-540-73521-2. (pages xv, 17, 18, 20, 22, 27).

[192] B. Wetzstein, Z. Ma, A. Filipowska, M. Kaczmarek, S. Bhiri, S. Losada, J.-M. Lopez-Cob, and L. Cicurel. Semantic business process management: A lifecycle based requirements analysis. In *SBPM*, 2007. (page 20).

[193] B. Wetzstein, Z. Ma, and F. Leymann. Towards measuring key performance indicators of semantic business processes. *Business Information Systems*, 7:227–238, 2008. (pages 22, 66).

[194] P. Wieder, J. Butler, W. Theilmann, and R. Yahyapour, editors. *Service Level Agreements for Cloud Computing*, volume 2506. Springer, 2011. ISBN 978-1-4614-1614-2. (pages 51, 52).

[195] L. Wu, S. K. Garg, and R. Buyya. Sla-based admission control for a software-as-a-service provider in cloud computing environments. *Journal of Computer and System Sciences*, 78(5):1280 – 1299, 2012. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2011.12.014. URL http://www.sciencedirect.com/science/article/pii/S0022000011001590. JCSS Special Issue: Cloud Computing 2011. (page 56).

[196] L. Wu, S. K. Garg, R. Buyya, C. Chen, and S. Versteeg. Automated sla negotiation framework for cloud computing. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 235–244, May 2013. doi: 10.1109/CCGrid.2013.64. (page 55).

[197] Y. Xiaoyong, T. Hongyan, L. Ying, J. Tong, L. Tiancheng, and W. Zhonghai. A competitive penalty model for availability based cloud sla. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 964–970, June 2015. doi: 10.1109/CLOUD.2015.142. (pages 132, 151).

[198] E. Yaqub, R. Yahyapour, P. Wieder, C. Kotsokalis, K. Lu, and A. I. Jehangiri. Optimal negotiation of service level agreements for cloud-based services through autonomous agents. In *2014 IEEE International Conference on Services Computing*, pages 59–66, June 2014. doi: 10.1109/SCC.2014.17. (page 55).

[199] W. ZhenHua, H. Yousen, D. ZiYun, and Z. Wei. Soa - bpm based information system for promoting agility of third party logistics. In *Automation and Logistics, 2009. ICAL '09. IEEE International Conference on*, pages 248–252, 2009. doi: 10.1109/ICAL.2009.5262920. (page 118).

[200] Q. Zhu and R. Fung. Design and analysis of optimal incentive contracts between fourth-party and third-party logistics providers. In *Automation and Logistics (ICAL), 2012 IEEE International Conference on*, pages 84–89, 2012. doi: 10.1109/ICAL.2012.6308175. (page 118).

[201] F. Zulkernine, P. Martin, C. Craddock, and K. Wilson. A policy-based middleware for web services sla negotiation. In *2009 IEEE International Conference on Web Services*, pages 1043–1050, July 2009. doi: 10.1109/ICWS.2009.157. (page 55).