A First Approach to Model SLAs for Composite Services, using WS-Agreement*

Carlos Müller, J. A. Parejo, A. Ruiz-Cortés

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

Abstract. When organizations began to use intensively services oriented applications (SOA) they arose the need of service level agreements (SLAs) to provide the confidence needed on quality of service (QoS). Nowadays, we have a consensus on what an SLA for a service is and as consequence, languages to specify SLAs as WS-Agreement have been proposed and they are becoming popular. One of the key topics in SOA research at present are composite services (CS). CS have been tackled by many authors and there is a consensus on their definition. However, not many works deal with SLAs for CS. The reason could be the lack of an intensive use of CS applications in business context. But in future CS applications comprised by heterogeneous services -defined in diverse specification languages or supported by different platforms- may gain the attention of companies and they will need a QoS support. Then, we consider interesting to obtain a general definition of SLAs for CS starting from current proposals. In this paper we analyze different meanings given in the literature for the expression "SLAs for CS", we find the shared aspects, we study the unshared ones and we propose to include important additional elements into our abstract model for defining SLAs for CS, such as temporal information and QoS properties about the whole CS. Finally, we promote the use of WS-Agreement with some non-intrusive extensions proposed by us for the establishment of SLAs for CS according to our abstract model, including several use cases.

1 Introduction

Services oriented applications (SOA) have been widely studied by many authors and now exists a consensus on the definition of services. Services are basically functional components with non-functional properties -also called quality of service (QoS) properties- offered by a provider in a distributed context. When organizations began to use intensively SOA, arose the need of a QoS assurance. In this context, the service level agreements (SLAs) were developed and nowadays there is a consensus on what an SLA for a service should include. That is:

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

(1) information about the agreement context as parties involved; (2) a group of terms including the offered/demanded operations -functional properties- and the guaranteed/required QoS; (3) other information as preferences, penalties, etc. As consequence, general purpose languages to specify SLAs as WS-Agreement (WS-Ag) [7] -a proposed recommendation of the Object Grid Forum, including a XML-based specification language for SLAs- have been proposed and they are becoming popular.

One of the most recent challenges in SOA are composites services (CS). CS have been tackled by many authors and most of them define a CS as a group of services working together to get a global functionality. The definition elements of a CS given in literature are: (1) services that compound the CS, (2) structural information about how component services are connected -also called topology or topological information-, (3) QoS properties of each component service (local QoS properties), (4) QoS properties of the whole CS (global QoS properties), usually depending on local QoS properties. Instead of this established ideas on CS, not many works deal with SLAs for CS. A possible reason could be the lack of an intensive use of CS applications in a business context. However, in future this kind of applications may gain the attention of SOA organizations -those that bet on SOA-. Consequently, it will be needed a management of several CS comprised of heterogeneous services defined in diverse specification languages and implemented in several platforms. Thus, we consider essential to get support of QoS assurance for CS. For instance: imagine the manager of a SOA company using a CS for getting travel information. If the CS fails, he needs an alternative to keep working. If an SLA for cited CS would have been established, it could have included a term which considers a CS failure.

Related works show that there are different shades of meaning on SLA for CS. So, we consider interesting to obtain an abstract definition of SLAs for CS supporting all of them. Then, we analyze in this paper the different meanings given in the literature for SLAs for CS and we include into our model: (1) the shared aspects such as: local QoS properties, local QoS terms, local SLAs -one SLA per component service-, and global QoS aggregated from local properties; (2) the unshared aspects such as: topology of CS and standard languages use; and (3) some elements slightly dealt in the literature as QoS attributes and terms of the whole CS, and temporality. Finally, we instantiate our model by implementations according to our abstract model. These implementations use WS-Ag with some non-intrusive extensions for the establishment of SLAs for CS -understanding non-intrusive as an extension which makes good use of extension point prefixed in the WS-Ag specification-.

The remainder is organized as follows. Section 2 include the comparative analysis of current proposals, new needs, and an abstract model for SLAs for CS. In section 3 we make an overview of WS-Ag. Section 4 studies the SLA for CS support by current WS-Ag and proposes some extension to completely cover our model. Section 5 shows two use cases, a black-box use case and a use case with all elements of our abstract model. Finaly, Section 6 presents conclusions and future work.

2 SLAs for Composite Services

The expression: "SLAs for CS" is commonly used in the related works, but with different shades of meaning. We have compared current proposals in this context and we have found that although most aspects of the SLAs for CS definitions are used in a similar way -as you can see on table 1-, there are some aspects whose meaning is not exactly the same for all authors. In 2.1 we include an analysis of the current status including shared and unshared aspects. In 2.2 we enunciate several needed elements of SLAs for CS that were laxly used in previous works. And in 2.3 we propose an abstract definition for SLAs for CS considering all mentioned aspects of SLAs for CS.

2.1 Current Status

Related work shows aspects with similar meaning concern to: (1) the component service of CS -local QoS properties, local QoS constraints, and local SLAs-; (2) the entire CS -global QoS properties, global QoS constraints, and a global SLA for the CS-. But it is important to highlight that these global aspects of the CS are defined generally in function of local aspects of component services. For instance a global QoS property as "cost for the CS" is calculated by an aggregation on the local QoS properties called "service cost". The same meaning is given for global constraints and global SLAs. Thus, current proposals understood them in relation with local constraints and SLAs.

However some few authors as [1] change the cited meaning of the global aspects to get more expressiveness in specific scenarios. For instance using a kind of global constraint by means of a called "service selection constraint" similar to: "if you choose a service of this provider, you must choose those others services from the same provider", which allows the expression of stateful services dependences. If we consider each service individually, that constraint does not makes sense. This and other new needs -such as temporality or topology information- in "SLAs for CS" are discussed in the following section.

2.2 New Needs

In [1] the authors consider important to use a global QoS constraints -concretely to define dependences between component services- which only makes sense related to a CS. So, this kind of global QoS constraint can be considered more abstract than global constraints that only makes sense as a function of local constraints. Moreover, we consider a more abstract SLA for the whole CS which could include that kind of global QoS constraints and in addition, global QoS properties related to the whole CS -not necessarily aggregated from properties of component services-. Thus we could work with SLAs for CS as now with SLAs for single services. It allows to conceive a CS+SLA as a black box; and for instance a manager could interchange a CS used in his company by another one without any idea of their component services.

The topological information of the CS is considered as an essential aspect in most of studied works: [1,2,5,4,6,9,12,13]. The topology of a CS include information about mutually exclusive execution paths and parallel execution paths generally. This kind of structural information is really useful for management of CS at runtime. Thus, we think that it is interesting to include in a global SLA information about the local SLAs, considering if a local SLA is mandatory or optionality and parallel or not. With this information the execution of a CS could be stopped if a mandatory term has not been fulfilled.

We consider temporality as another key aspect of an SLA for CS. In a previous work [8], we proposed a temporal domain specific language which can be used to express validity periods over the entire SLA or many of their components such as terms or preferences. Increasing the level of abstraction we could use the same proposal to express validity periods in SLA for CS and their elements, giving more expressiveness to the SLA. With this temporal awareness, we could express a validity period for:

- The whole CS
- Any set of terms from global or local SLAs -it allows to express temporal dependencies between different component services. For instance two parallel component services could have terms with the same validity period, determining the parallelism-.
- Preferences over any set of terms from local or global SLAs -it allows to express preferences in different periods. For instance in a validity period an optional local SLA could be the best option for user, but in other period the same local SLA could be the last option-.

Finally we consider using standard specifications for the SLAs for CS as a good practice. Only three of the studied proposals [6,9,10] include into their works support for standard SLA languages. [6] and [10] use WSLA -one of the predecessor of WS-Ag- and [9] uses Ws-Ag. The rest of studied works use ad hoc specifications for expressing SLAs or BPEL to model the execution process of the CS. In this paper we promote the use of WS-Ag with some extensions as a possible general specification language for the abstract model discussed in next section.

Proposals	Our Proposal.	Dyachuk & Deters [6]	Tomarchio et al. [9]	Charfi et al. [4]	Baligand et al. [2]	Di Penta et al. [5]	Cardellini et al. [3]	Ardagna & Pernici [1]	L. Zeng et al. [13]	Narendra et al. [10]	Thißen & Wesnarat [12]
Local QoS Properties	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Local QoS Constraints	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Local SLAs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Global QoS Properties	✓	∼	∼	∼	∼	∼	∼	∼	∼		∼
Global QoS Constraints	✓	∼		∼	∼	∼	∼	✓	∼	∼	
Global SLA	✓			∼	∼	∼	∼	∼		∼	∼
Topology of CS	✓	✓	✓	✓	✓	✓			✓		✓
Temporality	✓	τ	τ				τ				
Standard Language	✓	Std	Std							Std	

✓=aspect included, ∼=aspect included as an aggregation of the locals.
τ=some temporal aspects naively included, Std=some standard language used.

Table 1. Included Aspects in the Definition of SLAs for **CS**

2.3 Abstract Definition

Considering the current status and new needs mentioned, we could express an abstract definition of SLA for CS as follows:

$$SLA_C = \{S, P_s, P_c, T_s, T_c, Ctx\}$$

Where:

- S is a set of component services;
- P_s is a set of local QoS properties;
- P_c is a set of global QoS properties -expressed in relation with the whole CS, or expressed as a function on the elements of P_s-;
- T_s is a set of terms related with a specific component service -it allows two kinds of terms: (1) temporal-aware service terms: defining the operations of the service, allowing validity periods; and (2) temporal-aware guarantee terms: including the QoS guaranteed or required by the parties, allowing validity periods-;
- T_c is a set of terms related to the whole CS, or in relation with the elements of T_s as a function -it also allows two kinds of terms: (1) temporal-aware composition terms: defining the CS operations and even topology, allowing validity periods; and (2) temporal-aware guarantee terms: including the QoS guaranteed or required by the parties in relation with the whole CS, allowing validity periods-;
- Ctx is a context which includes information related to the SLA such as: a global validity period for the whole CS, the mandatory SLAs list, the parties involved and their roles or any other information.

Figure 1 denotes the mentioned elements of our abstract definition, showing a composite service with three component services representing global and local QoS properties by means of connected rhombus. Figure includes local and global SLAs with a generic WS-Agreement structure which will be explained in the following section. Then in local SLAs it is shown by simplicity only a set of terms (ST in figure) and a set of guarantee terms (GT in figure) defined on QoS properties. The global SLA (SLA_C in figure) is comprised of the mentioned new needs for CS (highlighted in a discontinuous square in figure), such as: (1) a set of temporal-aware composition terms giving the information about the whole CS, (2) a set of guarantee terms defined on the global QoS properties, (3) a context supporting a global validity period definition for the whole CS. In addition, the SLA_C allows to specify the information included in local SLAs by means of aggregations of its service terms and guarantee terms defined as a function on the local QoS properties (showed at bottom of terms in figure).

3 WS-Agreement in a nutshell

WS-Ag specifies an XML-based language and a protocol for advertising the capabilities of service providers, creating agreements based on agreement offers.

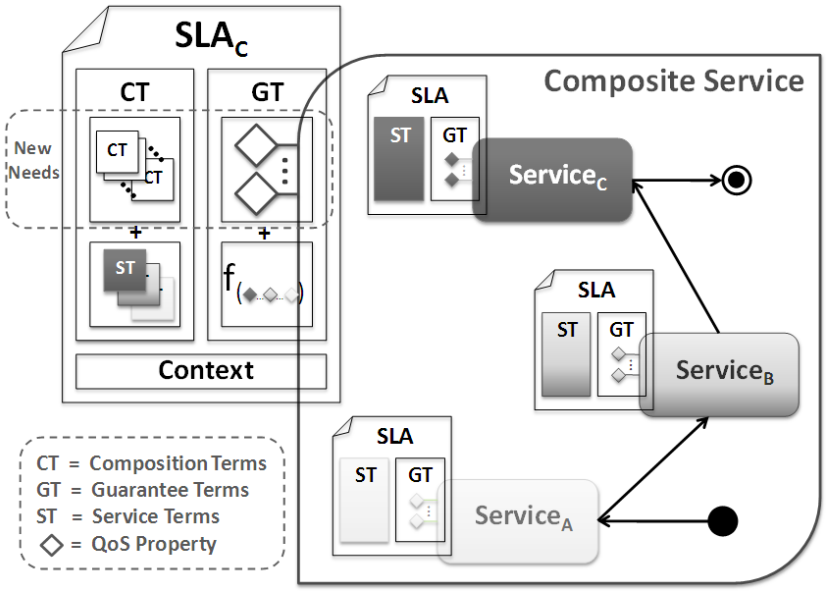

Fig. 1. Elements of SLAs for Composite Services.

The structure of an agreement for WS-Ag is comprised of: (1) **Name**: identifies the agreement and can be used for reference. (2) **Context**: it includes information such as the name of the parties and their roles of initiator or responder of the agreement. Additionally, it can refer to an agreement template if needed. In this element, an agreement lifetime can be defined by means of an element called "ExpirationTime". (3) **Terms**: agreement terms are wrapped by term compositors, which allow simple terms or sets of terms to be denoted by "ExactlyOne", "OneOrMore", or "All". The two main types of terms are: (a) *Service terms*: they provide information to instantiate or identify services and operations involved in the agreement. Additionally, it can comprise of information about the measurable service properties. (b) *Guarantee terms*: they describe the service level objectives (SLO) agreed by the parties. It also includes the scope of the term (e.g. a certain operation of a service or the whole service itself); a "QualifyingCondition" that specifies the validity conditions under which the term is applied; and information about business properties in the "BusinessValueList" element such as "Importance", "Penalty" or "Reward" and "Preference".

In order to create agreements, WS-Ag allows us to specify templates with the above structure, but including agreement "Creation Constraints" that describe the variability allowed by a party and it should be taken into account during the agreement creation process. In a template the initiator can leave elements empty to negotiate it later.

4 WS-Agreement to Specify SLAs for Composite Services

Current WS-Ag specification allows the definition of SLAs for a set of services by means of:

- Several variables into the "service properties" element -to include local QoS properties-;
- Several "service description terms" -one per component service of the CS, at least-;
- Several "guarantee terms" referring to one or more service description terms and including the QoS guaranteed or required by the parties;
- Some of the mentioned context information as the information about the parties involved and their roles could be included in the context of WS-Ag.

With these allowed aspects WS-Ag only has completely covered some elements of the previous abstract definition, concretely the underlined here:

$SLA_C = \{\underline{S}, \underline{P_s}, P_c, \underline{T_s}, T_c, Ctx\}$. To cover P_c and T_c we would need a way for specifying global QoS properties, CS guarantee terms and CS description terms, allowing temporal and topological information; all of them not necessary computed as function of local aspects of component services. And to cover Ctx we would need a way for specifying temporal information over the entire CS and the mandatory SLAs list.

A study of the temporality allowed by current WS-Ag specification is included in [8]. Basically WS-Ag include an expiration time for the SLAs and it only allows a disjoint period of time; and to express temporality on terms.

We propose to use some non-intrusive extensions for WS-Ag to tackle the problem of specifying the cited uncover aspects. We consider their non-intrusive because we do not change the current WS-Ag specification structure. We make good use of different extension points established in WS-Ag, concretely:

- To specify temporal information, we promote the use of the improvement of temporal-awareness for WS-Ag developed in [8] but applied now to SLAs for CS. Thus we could specify: (1) a global validity period for the entire SLA for CS using the context element of WS-Ag; and (2) validity periods for any term -allowing to determine the parallel execution on any set of component services, as we show later in Section 5- or any preference definition -allowing to express temporal-aware preferences over the optional component services, as we show later in Section 5- using the qualifying condition element of WS-Ag.
- CS guarantee terms can be expressed as guarantee terms of WS-Ag scoping to the whole CS -or a part-;
- To specify CS properties and CS description terms we propose two alternatives such as: (a) to use service properties and service description term, both elements from current WS-Ag -which implies to change the original meaning of those items, because we add properties and description term related to the CS, not to a single service-; or (b) to create another term type called "CS Term" with two subtypes called "CS Properties" and "CS Description

Term" -That second alternative is our selected for the use cases of Section 5 because it does not change the meaning of service terms from current WS-Ag as occurs in first-.
- To include topology information inside the CS description terms we will use: (1) Validity Periods inside guarantee terms, scoping to the concrete part of the CS described in a CS description term. That periods will determine which operations are parallel or optional -if two validity periods for operations have temporal instants in common-; (2) the compositor term elements of WS-Ag are used to join different set of mandatory operations which contain one optional operation at least.

5 Use Cases

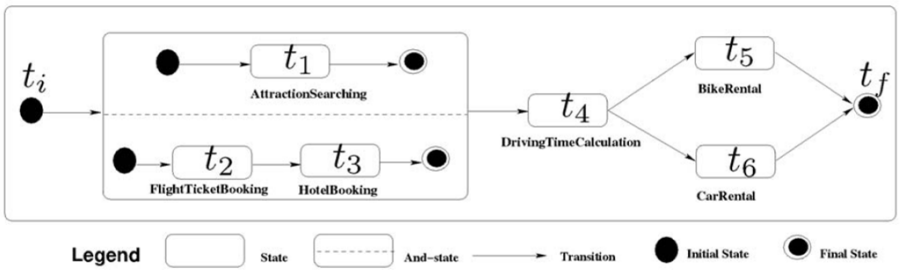

Fig. 2. Travel-planner composite service of Zeng et al [13].

We base our case study on the "Travel-Planner" of zeng et al. [13], showed in Figure 2, because it is commonly used in literature: [5,3]. However, to validate our model we include some additional information or constraints, such us:

- The number of distinct provider for the CS must be less than 4.
- A list of mandatories SLAs is included in context to make easier to invalidate an agreement if any agreement term of a mandatory SLA is violated.
- CSDTs showing each CS operation as elements which include the SDT name of the service which perform the operation -coulding be the same SDT for different operations-.
- The optionality is solved by means of preferences defined at validity periods.
- Topological information included:
  - Parallelism is included with the validity periods of each operation inside guarantee terms.
  - Optionality is included by means of two CSDTs including inside each one optional operation -separated by compositor elements of WS-Ag-.

It is important to highlight again that to specify the validity periods, we stand on the proposal made for us in [8]. So, in our study cases we will use the elements "GlobalPeriod" (GP) and "QualifyingCondition" (QC), to define the validity periods for the whole agreement and for concrete agreement parts. But, for simplicity, we will use natural expressions to define the concrete intervals comprising the validity periods.

A "Black-Box" study case -without any topological information nor SDT name fixed- is denoted in Figure 3. The key of this study case is the independent definition of a CS by the initiator party of an agreement. To define the CS it is not necessary to include concrete providers for each operation, it only include the constraint on the number of distinct provider. So, in future the initiator will be able of use the same template with different providers.

```
<Template>

     <Context>
          <GP>...</GP> <!-A global validity period definition-->
          <MandatorySLAsList>...</ MandatorySLAsList>
     </Context>

     <All>
          <CSDT Name='TravelPlanner'>
               <AttractionSearchingSDTName>     </AttractionSearchingSDTName>
               <FlightTicketBookingSDTName>     </FlightTicketBookingSDTName>
               <HotelBookingSDTName>            </HotelBookingSDTName>
               <DrivingTimeCalculationSDTName>  </DrivingTimeCalculation>
               <BikeRentalSDTName>              </BikeRentalSDTName>
               <CarRentalSDTName>               </CarRentalSDTName>
               <DistinctProv>                   </DistinctProv>
          </CSDT>

          <CSProperties>
               DistinctProv (Location \CSDT\DistinctProv)
          </CSProperties>

          <GT Name='ProviderLimit'>
               <Scope>   CSDT  </Scope>
               <SLO>   DistinctProv < 4   </SLO>
          </GT>
      </All>

</Template>
```

Fig. 3. Black-Box WS-Agreement Template.

In the following example we have included a more complete study case with the following topological embedded information and SDTs fixed:

1. Topological info:
   - Optionality in transport rental (with two CSDTs with the same validity period definition). This optionality is solved by means of preferences inside the agreement. You can see how the bike rental operation is chosen when sun is shining and the car rental is chosen at nights.

  - Parallelism in attraction searching and bookings (with the same validity period definition)

2. Some SDT Fixed:
   - We can fix any part of SDT: (Provider, Cost, ...). In this case we have fixed the provider of some CS operations.

```
<Template>

    <Context>
        <GP>...</GP> <!-A global validity period definition-->
        <MandatorySLAsList>...</ MandatorySLAsList>
    </Context>

    <All>
        <Exactlyone>
           <CSDT Name='BikeCSDT'>
              <AttractionSearchingSDTName>        </AttractionSearchingSDTName>
              <FlightTicketBookingSDTName>    t2  </FlightTicketBookingSDTName>
              <HotelBookingSDTName>           t3  </HotelBookingSDTName>
              <DrivingTimeCalculationSDTName>     </DrivingTimeCalculation>
              <BikeRentalSDTName>             t5  </BikeRentalSDTName>
              <DistinctProv>                      </DistinctProv>
           </CSDT>
           <CSDT Name='CarCSDT'>
              <AttractionSearchingSDTName>        </AttractionSearchingSDTName>
              <FlightTicketBookingSDTName>    t2  </FlightTicketBookingSDTName>
              <HotelBookingSDTName>           t3  </HotelBookingSDTName>
              <DrivingTimeCalculationSDTName>     </DrivingTimeCalculation>
              <CarRentalSDTName>              t6  </CarRentalSDTName>
              <DistinctProv>                      </DistinctProv>
           </CSDT>
        </Exactlyone>

        <SDT Name='t2'>
            <Provider>    "Amadeus"    </Provider>
            <Cost>           ...       </Cost>
        </SDT>

        <SDT Name='t3'>
            <Provider>    "Amadeus"    </Provider>
            <Cost>           ...       </Cost>
        </SDT>

        <SDT Name='t5'>
            <Provider>    "RentABike"    </Provider>
            <Cost>           ...       </Cost>
        </SDT>

        <SDT Name='t6'>
            <Provider>    "CarRentalCorp"    </Provider>
            <Cost>           ...       </Cost>
        </SDT>

        <CSProperties>
          <Variables>
            ...DistinctProv (Location \CSDT\DistinctProv)
            ...BikeRentalSDTName (Location \CSDT\BikeRentalSDTName)
            ...CarRentalSDTName (Location \CSDT\CarRentalSDTName)
            ...AttractionSearchingSDTName (Location \CSDT\AttractionSearchingSDTName)
            ...FlightTicketBookingSDTName (Location \CSDT\FlightTicketBookingSDTName)
            ...HotelBookingSDTName (Location \CSDT\HotelBookingSDTName)
          </Variables>
        </CSProperties>

        <GT Name='ProviderLimit'> <!-not a distinct provider for each service-->
            <Scope>   CSDT   </Scope>
```

```
     <SLO>   DistinctProv < 4   </SLO>
</GT>

<GT Name='OptionalTransportValidityPeriod'>
     <QC>  <!-Optional validity period definition-->  </QC>
     <Scope>
          CSDT\BikeRentalSDTName
          CSDT\CarRentalSDTName
     </Scope>
     <SLO>
          (BikeRentalSDTName <> null) and (CarRentalSDTName <> null)
     </SLO>
</GT>

<GT Name='AttractionSearchingValidityPeriod'> <!-parallel with 2 next GTs-->
     <QC>  <!-Attraction validity period definition-->
          Jan, 1st 2009:09:00  -  Jan, 1st 2009:09:05
     </QC>
     <Scope>
          CSDT\AttractionSearchingSDTName
     </Scope>
     <SLO>  (AttractionSearchingSDTName <> null)  </SLO>
</GT>

<GT Name='FlightBookingValidityPeriod'>
     <QC>  <!-Flight Booking validity period definition-->
          Jan, 1st 2009:09:01  -  Jan, 1st 2009:09:02
     </QC>
     <Scope>
          CSDT\FlightTicketBookingSDTName
     </Scope>
     <SLO>  (FlightTicketBookingSDTName <> null)  </SLO>
</GT>

<GT Name='HotelBookingValidityPeriod'>
     <QC>  <!-Hotel Booking validity period definition-->
          Jan, 1st 2009:09:03  -  Jan, 1st 2009:09:04
     </QC>
     <Scope>
          CSDT\HotelBookingSDTName
     </Scope>
     <SLO>  (HotelBookingSDTName <> null)  </SLO>
</GT>

<GT Name='BikeRentalPreference'>
     <QC>  <!- BikeRentalPreference validity period definition-->
          "When Sun Shining" : (periodical interval)
     </QC>
     <SLO>  (BikeRentalSDTName <> null)  </SLO>
     <BVL>
          <Preference> <!-original WS-Ag preference (only a utility value)-->
               <SDTReference>  BikeCSDT  </SDTReference>
               <Utility>  1.0  </Utility>
               <SDTReference>  CarCSDT  </SDTReference>
               <Utility>  0.0  </Utility>
          </Preference>
     </BVL>
</GT>

<GT Name='CarRentalPreference'>
     <QC>  <!- CarRentalPreference validity period definition-->
          "At nights" : (periodical interval)
     </QC>
     <SLO>  (BikeRentalSDTName <> null)  </SLO>
     <BVL>
          <Preference> <!-original WS-Ag preference (only a utility value)-->
               <SDTReference>  BikeCSDT  </SDTReference>
               <Utility>  0.0  </Utility>
```

```
                    <SDTReference>  CarCSDT  </SDTReference>
                    <Utility>  1.0  </Utility>
                </Preference>
            </BVL>
        </GT>
    </All>
</Template>
```

6 Conclusion and Future Work

An abstract model for SLAs for CS is needed, specially on management of CS -including activities such as: monitoring, binding of component services, orchestrating heterogeneous component services defined in different specification languages or stored in diverse platforms. Thus, in this paper we have established a novel abstract model for SLAs for CS. We also propose an instantiation of our model, exposing two alternative implementations with WS-Ag. In order to do so, we make good use of WS-Ag extension points and other temporal extension proposed in a previous work [8]. Once developed a general framework with our abstract model, in future works we will develop an advance management of CS taking into account the mentioned new needs for task such as: the SLA monitoring, the SLA renegotiation, etc. And finally, we plan to adapt to CS a previously presented proposal of services binding using constraint satisfaction problems. [11].

References

1. D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *Software Engineering, IEEE Transactions on*, 33(6):369–384, 2007.
2. Fabien Baligand, Nicolas Rivierre, and Thomas Ledoux. A declarative approach for qos-aware web service compositions. pages 422–428. 2007.
3. V. Cardellini, E. Casalicchio, V. Grassi, and F. Lo Presti. In *IW-SOSWE'07: 2nd international workshop on Service oriented software engineering*, New York, NY, USA.
4. Anis Charfi, Rania Khalaf, and Nirmal Mukhi. Qos-aware web service compositions using non-intrusive policy attachment to bpel. pages 582–593. 2007.
5. M. Di Penta, G. Canfora, G. Esposito, V. Mazza, and M. Bruno. In *GECCO'07: Proceedings of the 2007 conf. on Genetic and evolutionary computation*, London, England, UK.
6. Dmytro Dyachuk and Ralph Deters. Using sla context to ensure quality of service for composite services. In *Pervasive Services, IEEE International Conference on*, pages 64–67, 2007.
7. OGF Grid Resource Allocation Agreement Protocol WG (GRAAP-WG). Web Services Agreement Specification (WS-Agreement) (v. gfd.107), 2007.
8. C. Müller, O. Martín-Díaz, A. Ruiz-Cortés, M. Resinas, and P. Fernández. Improving Temporal-Awareness of WS-Agreement. In *Proc. of the 5^{th} International Conference on Service Oriented Computing (ICSOC)*, pages 193–206, Vienna, Austria, Sept 2007. Springer Verlag.

9. G. Di Modica, O. Tomarchio, and L. Vita. A framework for the management of dynamic SLAs in composite service scenarios. In *Proc. of the 1st Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC'07)*, Vienna, Austria, Sept 2007. Springer Verlag.
10. N. Narendra, Karthikeyan Ponnalagu, Jayatheerthan Krishnamurthy, and R. Ramkumar. Run-time adaptation of non-functional properties of composite web services using aspect-oriented programming. pages 546–557. 2007.
11. A. Ruiz-Cortés, O. Martín-Díaz, A. Durán, and M. Toro. Improving the Automatic Procurement of Web Services using Constraint Programming. *Int. Journal on Cooperative Information Systems*, 14(4), 2005.
12. Dirk Thißen and Pimjai Wesnarat. Considering qos aspects in web service composition. pages 371–377.
13. L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.